VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Klient webových služeb s dynamickým GUI pro mobilní telefony

Web service client with dynamic GUI for mobile devices

2010 **Marek Rusnák**

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

. . . . . .

7.5.2010 Marek Rusnák

Chcem poďakovať svojmu konzultantovi
Ing. Pavlovi Moravcovi, Ph.D. za odborné konzultácie, ako aj cenné rady a pripomienky pri písaní tejto práce.

Abstrakt

V poslednej dobe došlo k veľkému rozšíreniu webových služieb, s vysokým potenciálom využitia, či už v komerčných, alebo nekomerčných oblastiach. Veľkou výhodou webových služieb je rýchle a jednoduché získavanie informácii rôzneho obsahu. Táto vlastnosť dostáva ešte väčší potenciál v kombinácii s pojmom mobilita. Práve kombinácia týchto dvoch vlastností je náplňou tejto bakalárskej práce, ktorej úlohou bolo vytvoriť aplikáciu s dynamickým GUI pre jednoduchšie typy webových služieb (WS), ktoré by popis WS interpretovalo priamo na mobilnom zariadení. V práci sú teoreticky popísané princípy fungovania WS a technológii, ktoré zaisťujú funkčnosť WS, základné pravidlá pre návrh a implementáciu efektívnych mobilných aplikácii, základné činnosti klienta WS, možnosti práce s dokumentmi formátu XML a vývojová platforma Java ME s modelom GUI pre MIDP 2.0 zariadenia. Praktickou časťou bakalárskej práce je návrh a implementácia klienta WS pre mobilné telefóny v programovacom jazyku Java.

Kľúčové slová: Webové služby, XML, XML Schéma, XML parser, WSDL, HTTP, SOAP, UDDI, Java ME, MIDP 2.0

Abstract

Recently, there has been an increase in using web services, with particularly high interest at commercial as well as in non-commertial areas. One of the greatest advantages of applying web services in practice is that information of different kind can be obtained in a quick and easy way. This advantage can have even better utilization in combination with the term mobility. And exactly combination of these two features is what this thesis is about: the main purpose was to create an application with dynamic GUI for simple types of web services (WS), which would interprate the characterization of WS directly on mobile device. This tesis also contains decsription of how WS and technologies, which are responsible for smooth operation of WS, theoretically operate; basic rules for scheme and implementation of effective mobile applications; basic actions of client operating the WS; working opportunities with XML format; development platform Java ME with model GUI for MIDP 2.0 devices. Practical part of this thesis is concerning project and implementation of WS client for mobile phones in Java programming language.

Key words: Web services, XML, XML scheme, XML parser, WSDL, HTTP, SOAP, UDDI, Java ME, MIDP 2.0

Zoznam použitých skratiek:

API – Aplication Programming Interface
CDC – Connected Device Configuration
CLDC – Connected Limited Device Configuration
DOM – Document Object Model
DTD – Document Type Definition
GCF – Generic Connection Framework
GUI – Graphical User Interface
HTTP – Hyper Text Transfer Protocol
IT – Information technology
JAR – Java Archive
Java ME – Java Micro Edition
Java EE – Java Enterprise Edition
Java SE – Java Standard Edition
MIDP – Mobile Information Device Profile
PC – Personal Computer
SOAP – Simple Object Access Protocol
UDDI – Universal Description Discovery and Integration
URL – Uniform Resource Locator
WBXML – WAP Binary XML
WML – Wireless Markup Language
WSDL – Web Service Definition Language
WSDWG – Web Services Description Working Group
WWW – World Wide Web
XML – Extensible Markup Language

Obsah

1 Úvod 2
2 Princíp a fungovanie webových služieb (WS) 3
2.1 XML ako základ webových služieb 3
2.2 Webové služby. 4
2.3 SOAP 5
2.4 WSDL 6
2.5 Prenos služby a SOAP komunikácia 7
2.6 UDDI 8
3 Vlastnosti s funkcie klienta WS 9
3.1 Efektívna mobilná aplikácia 9
3.2 Klient WS 10
4 Návrh implementácie 13
4.1 Návrh funkčnej časti klienta 13
4.2 Návrh grafického užívateľského rozhrania 15
4.2.1 Základné menu GUI 15
4.2.2 GUI SOAP komunikácie 16
4.2.3 Typický beh aplikácie 17
5 Implementácia 19
5.1 Použité technológie 19
5.1.1 Java ME 19
5.1.2 XML parser 21
5.2 Implementácia balíčkov 22
5.2.1 Balíček main_gui 23
5.2.2 Balíček http_communication 25
5.2.3 Balíček wsdl_parser 27
5.2.4 Balíček service 28
5.2.5 Balíček gui_ws 31
5.2.6 Balíček soap_parser 31
6 Testovanie 33
7 Záver 37
Zoznam použitej literatúry 38
Zoznam príloh

1 Úvod

V rebríčku široké spektra oblastí informačných technológií zastávajú vedúcu pozíciu nepochybne internetové technológie, založené na globálnom systéme navzájom prepojených počítačových sietí , nazývanom Internet. Internet je všeobecne veľmi rozšíreným informačným médiom umožňujúci distribúciu a získavanie informácií a dát, akéhokoľvek typu. Myslím si, že môžeme povedať, že na Internete je možné nájsť s trochou úsilia a vytrvalosti naozaj všetko. V skutku je toto médium naozaj veľmi silným poskytovateľom informácii, či už bežného alebo odborného obsahu a väčšina z nás si bez Internetu nedokáže predstaviť pracovný ani osobný život. Avšak pri hľadaní konkrétnej informácie často nastáva problém, spôsobený práve gigantizmom Internetu, ako informačného média. Z tohto dôvodu býva zvyčajne výsledkom vyhľadávania informácie na Internete veľké množstvo informačných zdrojov, ktoré je nie vždy jednoduché prezrieť a dostať sa k hľadanej a pokiaľ možno aktuálnej informácii. Tento problém do istej miery riešia webové služby, ktoré automatizujú prístup k informáciám akéhokoľvek typu. Cieľom technológie webových služieb je poskytovanie aktualizovaných informácii užívateľom bez zložitého vyhľadávania. Užívateľ Internetu, tak môže za krátku dobu a jednoduchým spôsobom získať hľadanú informáciu z akéhokoľvek zariadenia nezávisle na operačnom systéme.

Bakalárska práca sa zaoberá princípom fungovania webových služieb a popisom jednotlivých technológii, ktoré spoločne WS vytvárajú. Praktickou úlohou práce je návrh a implementácia klienta WS určeného pre mobilné zariadenia, ktorý bude schopný genericky spracovávať jednoduché WS. Užívateľ, tak získa pomocou klienta WS na mobilnom zariadení rýchly prístup k mnohým informáciám poskytovaných WS.

Práca je rozdelená do siedmich kapitol, kde prvá a posledná kapitola sú úvod a záver. Druhá a tretia kapitola sa zaoberajú teoretickou problematikou jednotlivých technológii WS, ako XML, SOAP, WSDL a popisom aplikácií pracujúcich s WS na strane klienta. Praktickej časti práce sa venujú ďalšie tri kapitoly, ktoré reprezentujú štandardný algoritmus tvorby aplikácii. Tretia kapitola sa zaoberá návrhom generického klienta WS, vo štvrtej kapitole je následne popísaná implementácia s použitými technológiami a v poslednej kapitole sú uvedené príklady WS spolu s popisom testovania výslednej aplikácie. V textových prílohách sa nachádza užívateľská príručka a triedne diagramy hlavných procesov aplikácií. Na priloženom CD je k dispozícii programátorská príručka spolu so zdrojovými kódmi aplikácie a inštalačné súbory.

2 Princíp a fungovanie webových služieb

Webové služby (WS) môžeme charakterizovať ako komunikáciu dvoch počítačov pomocou webového rozhrania, kde jeden počítač je poskytovateľom webovej služby a druhý počítač je klientom, ktorý webovú službu využíva. Webové služby sú platformne nezávislá technológia fungujúca na princípe klient-server s vysokým potenciálom využitia, či už v komerčných, alebo nekomerčných oblastiach. Hlavným prínosom webových služieb je prístup klienta k potrebným informáciám z akéhokoľvek zariadenia (napr.: stolový počítač, notebook, PDA, alebo mobilný telefón), bez obmedzenia na použitom operačnom systéme, programovacom jazyku, prípadne i type procesoru.

2.1 XML ako základ webových služieb

Základom technológie webových služieb, na ktorom je založený popis WS i samotná komunikácia serveru s klientom je jazyk XML (Extensible Markup Language). XML je momentálne najsľubnejším značkovacím jazykom pre uchovávanie a výmenu informácii, nielen medzi aplikáciami, ale aj medzi aplikáciou a užívateľom. XML je platformne nezávislí, veľmi rozšírený súbor pravidiel tvorby textových formátov, ktoré umožňujú usporiadanie dát do štruktúr. Primárne bol jazyk XML určený pre výmenu informácii prostredníctvom WWW, dnes však našiel využitie aj v iných odvetviach informačných technológii, ako napríklad v databázových systémoch apod.

XML bol definovaný pracovnou skupinou W3C, ako zjednodušená podoba staršieho jazyka SGML (Standard Generalized Markup Language).

Text uchovávaný v XML dokumente je rozdelený pomocou značiek (tzv. TAG) na jednotlivé elementy. Základným pravidlom je, že značky musia byť vždy párové, čiže pre počiatočnú značku musí vždy existovať aj jeho ukončovacia značka. Výnimku tvoria akurát prázdne elementy, ktoré neobsahujú žiadne dáta. Značky prázdnych elementov však musia byť ukončené znakmi „/>“. Každý element ďalej môže, ale nemusí obsahovať vlastné atribúty, ktoré sa nachádzajú v počiatočnej značke a sú reprezentované dvojicou názov - hodnota. Medzi najhlavnejšie vlastnosti jazyka XML patrí jeho zrozumiteľnosť a prenositeľnosť. Z tohto dôvodu musí dokument napísaný pomocou jazyka XML splňovať určité pravidlá, aby bol zrozumiteľný a jasný pre ostatné aplikácie. Tieto pravidlá sa zapisujú, buď podľa historicky staršieho DTD (Document Type Definition), alebo podľa XML Schémy. V technológii webových služieb sa stretneme s XML Schémou, preto popis DTD preskočíme.

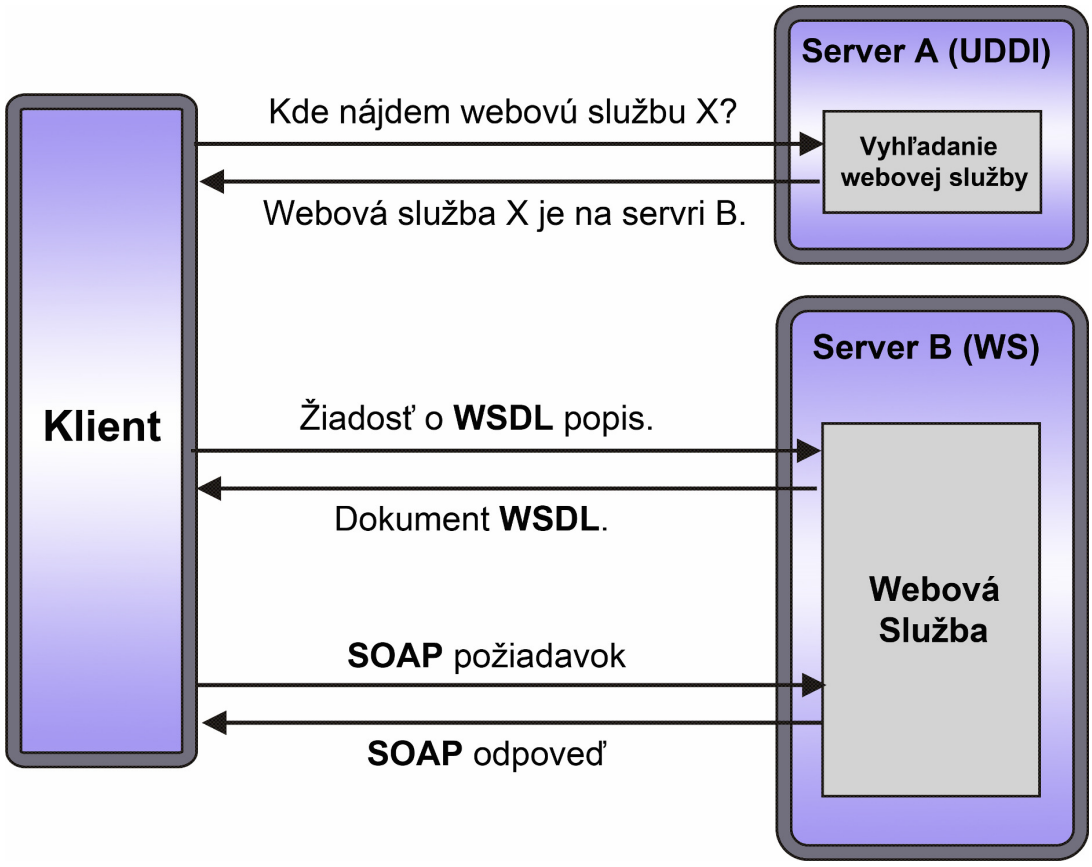
XML Schéma je dokument, ktorý definuje obsah a štruktúru tried dokumentov XML. Konkrétne, schéma XML popisuje elementy a atribúty, ktoré sa môžu nachádzať v danom dokumente a spôsob, akým majú byť elementy rozvrhnuté v hierarchickej štruktúre dokumentu. XML Schéma je oproti DTD sofistikovanejšia, poskytuje väčšiu úroveň vymedzenia, štruktúry tried dokumentov a je popísaná pomocou štandardnej syntaxe XML [1].

2.2 Webové služby

Ako už bolo spomenuté webové služby fungujú na princípe klient-server, kde server je poskytovateľom webovej služby (angl. Service provider) a klient (angl. Service requestor) využíva funkčnosť webovej služby a získava potrebné informácie. Tretím rozširujúcim komponentom webových služieb je tzv. register webových služieb (angl. Service registry). Na vzájomnej komunikácii klienta, poskytovateľa a registru je založená celá technológia webových služieb.

Webové služby sú v poslednej dobe veľmi využívanou technológiou práve vďaka softvérovej a platformnej nezávislosti. Aby však bolo možné zabezpečiť túto nezávislosť, musí byť výmena dát medzi aplikáciami, prípadne systémami striktne popísaná protokolmi a štandardmi. Protokoly popisujúce webové služby, ktoré úzko súvisia s tromi hlavnými komponentmi webových služieb, môžeme rozdeliť do štyroch skupín [2]:

- SOAP (Simple Object Access Protocol) - komunikácia medzi klientom a serverom pomocou XML správ
- WSDL (Web Services Definition Language) – štandardný jazyk pre popis rozhrania webových služieb
- UDDI (Universal Description, Discovery and Integration) – štandard pre registráciu a vyhľadávanie webových služieb.
- Prenos služby pomocou komunikačného protokolu HTTP



Obrázok 1: Komunikácia klienta s webovou službou na základe protokolov WSDL a SOAP.

Celý mechanizmus vzájomnej spolupráce hore uvedených protokolov a štandardov je znázornený na obrázku 1. V praxi to vyzerá tak, že užívateľ so záujmom využitia danej webovej služby si v registri webových služieb UDDI vyhľadá požadovanú webovú službu, prípadne pozná URL adresu WS. Odoslaním požiadavky získa WSDL popis rozhrania vybranej webovej služby. Z informácii, ktoré mu poskytne WSDL popis je užívateľ schopný komunikovať s webovou službou prostredníctvom správ protokolu SOAP.
V ďalších kapitolách si v krátkosti popíšeme jednotlivé protokoly a štandardy. Komunikačné protokoly pre prenos webových služieb nie sú priamou súčasťou technológie webových služieb, nakoľko však zabezpečujú ich prenos a komunikáciu, je vhodné v krátkosti ich predstaviť.

2.3 SOAP

V súčasnosti najperspektívnejším protokolom technológie webových služieb je protokol SOAP (angl. Simple Object Access Protocol). Ide o protokol popisujúci štruktúru informácii, ktoré sú posielané medzi klientom a poskytovateľom webových služieb. SOAP môžeme charakterizovať, ako protokol pracujúci na princípe volania vzdialených procedúr (RPC), prenášajúcich dáta v XML formáte. V princípe ide o jednosmerný prenos informácie medzi tzv. SOAP uzlami, od SOAP odosielateľa k SOAP príjemcovi, ktorého základom sú správy vo formáte jednoduchých XML dokumentov s koreňovým elementom Envelope.

Historicky je protokol SOAP prvým a hlavným protokolom webových služieb na rozdiel od ostatných protokolov a štandardov, ktoré boli definované neskôr. S prvotnou myšlienkou vytvoriť takýto komunikačný protokol prišla v roku 1998 firma Microsoft, a od počiatku bol protokol označený ako protokol SOAP. Už v roku 2000 bol protokol vo verzii 1.1 navrhnutý do registrácie W3C, avšak až jeho verzia 1.2 sa stáva 24. júna 2003 odporúčaným protokolom pod záštitou W3C.

Po krátkom predstavení a historickom úvode si ukážeme základné elementy a atributy SOAP správ. Správa protokolu SOAP je jednoduchý XML dokument, ktorý je zložený zo štyroch základných elementov:

- element ENVELOPE
- element HEADER
- element BODY
- element FAULT

Tieto štyri základné elementy rozdeľujú SOAP správy do štandardizovanej štruktúry dokumentu (Obrázok 2). Najhlavnejším a najdôležitejším elementom SOAP správy je element ENVELOPE, ktorým sme schopný identifikovať XML dokument a určiť, že ide o správu protokolu SOAP.

Ďalšie dva elementy HEADER a BODY sú potomkami elementu ENVELOPE, pričom element HEADER je nepovinný a nemusí sa v SOAP správe nachádzať. Prítomnosť elementu HEADER je daná poskytovateľom webovej služby a obsahuje údaje hlavičky.

Ďalším zo základných elementov SOAP správ je element BODY, ktorý obsahuje informácie o volaniach a odpovediach, najčastejšie v podobe ďalších elementov. Element BODY je v podstate jednoduchý element, ktorý obsahuje aktuálnu správu na základe popisu WSDL, pre definitívny koncový bod, ktorému je správa určená. V praxi je element BODY kľúčovým elementom komunikácie s webovou službou.

Posledným elementom je element FAULT obsahujúci chybové hlásenia a informácie o stave. Prítomnosť elementu FAULT v SOAP správe je voliteľná. Ak je však element v správe prítomný, musí byť jednoznačne potomkom elementu BODY a môže sa v správe nachádzať len jeden krát [3].

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org3/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
    ...
    </soapHeader>

    <soap:Body>
    ...
        <soap:Fault>
        ...
        </soap:Fault>
    </soap:Body>

</soap:Envelope>
```

Obrázok 2: Základná kostra SOAP správy obsahujúca všetky štyri základné elementy (Envelope, Header, Body, Fault)

2.4 WSDL

Ďalším základným komponentom webových služieb je štandard WSDL (angl. Web Services Definition Language). WSDL je jazyk vychádzajúci zo štandardu XML určený pre popis webových služieb. WSDL popis poskytuje tri základné informácie. Informácie o umiestnení webovej služby, akými protokolmi môžeme s webovou službou komunikovať a aké metódy s akými parametrami služba poskytuje. Na základe získaných informácii sme schopný, či už manuálne alebo automatizovane, generovať SOAP správy a grafické užívateľské prostredie (GUI) [4].

Prvá verzia WSDL 1.1 vznikla zlúčením troch jazykov firiem IBM, Microsoft a Arima s názvami NASSL, SCL a SDL a od počiatku bola prijatá konzorciom W3C za štandard. V súčastnosti pracuje pracovná skupina WSDWG W3C konzorcia na novšej verzii WSDL 2.0 [5].

WSDL dokument je jednoduchý dokument XML formátu, ktorý je rozdelený do danej štruktúry vzájomne previazaných elementov. Štandardným obsahom WSDL dokumentu je element definícia (angl. definitions), ktorý obsajhuje päť základných elementov:

- element types
- element message
- element portType
- element binding
- element service

Prvým elementom previazanej štruktúry WSDL dokumentu je element služba (angl. service), ktorý obsahuje jednu, alebo viac brán (angl. port). Každej bráne je pridružená väzba (angl. binding), ktorá určuje spôsob ako bránu volať. Element väzby definuje formát správy a detaily protokolu pre každý port. Obsahuje atributy meno (angl. name) a typ (angl. type), ktorý je referenciou na príslušné rozhranie (angl. portType) WSDL dokumentu. Rozhranie je ďalším a viac menej aj najhlavnejším elementom popisu WSDL, nakoľko definuje prístupný bod k webovej službe a popisuje operácie (angl. operation), ktoré webová služba poskytuje a im priradené správy. Štandard WSDL definuje štyri typy operácii v rozhraní. Na základe použitého typu je daný aj počet správ v dokumente. V praxi sa najčastejšie stretneme s typom operácie požiadavok-odpoveď (angl.request-response), zmienime sa však v krátkosti o všetkých typoch operácii:

- one-way operácia– operácia môže príjmať požiadavok, ale nevracia odpoveď
- request-response operácia – operácia môže príjmať požiadavok a vrátiť odpoveď
- solicit-response operácia – operácia môže poslať požiadavok a príjmať spätnú odpoveď
- notification operácia– operácia poslať správu, ale nečaká spätnú odpoveď

Správa (angl. message) je ďalším odvodeným elementom WSDL popisu, ktorý je odkazovaný z elementu operácie rozhrania. Posledným elementom je element typy (angl. types), ktorý popisuje štruktúru a dátové typy používané webovou službou. Pre maximálnu platformnú neutralitu využíva WSDL k definícii dátových typov syntaxi XML schémy. V XML schéme môžeme definovať štandardizované dátové typy podľa WSDL, ako napríklad reťazce, čísla s pohyblivou a pevnou čiarkou, pravdivostné hodnoty apod. Taktiež je možné definovať pevne definované dátové typy (simpleType) a zložené dátové typy (complexType), zostavené zo štandardných dátových typov, zo zložených dátových typov, prípadne ich kombináciou.

Z tohto textu je jasné, prečo sa popis WSDL považuje za jeden z hlavných komponentov technológie webových služieb. Bez jeho prítomnosti by bolo veľmi ťažké odhaliť a odhadnúť operácie, ktoré poskytuje webová služba a spôsob komunikácie.

2.5 Prenos služby a SOAP komunikácia

Komunikácia v technológii webových služieb je zabezpečená v drvivej väčšine prípadov jedným z najdôležitejších a najvyužívanejších komunikačných protokolov určených pre prenos dát medzi serverom a klientom, protokolom HTTP (angl. Hyper Text Transfer Protocol). HTTP je

jednoduchý protokol implementujúci malú množinu príkazov, schopný prenášať dáta určené URL adresou [6].

Vzájomná komunikácia HTTP klienta a HTTP serveru vypadá tak, že HTTP klient požiada o pripojenie HTTP server. Po vytvorení a potvrdení spojenia prostredníctvom protokolu TCP môže klient začať vysielať HTTP správy serveru. Server spracuje žiadosť klienta a vysiela odpoveď opäť vo forme HTTP správy. Každá odpoveď obsahuje stavový kód, ktorý indikuje stav požiadavku (úspech/neúspech).

Pri komunikácii v oblasti webových služieb, konkrétne pri komunikácií prostredníctvom protokolu SOAP s komunikačným protokolom HTTP je nutné brať ohľad na jednu skutočnosť. Pri komunikácií totiž nestačí len odoslať samotnú SOAP správu formátu XML prostredníctvom HTTP, ale je nutné prednastaviť potrebné záhlavia žiadosti HTTP. Jedným z týchto záhlaví je záhlavie Content-Type, ktoré popisuje spôsob zakódovania dát v tele správy a druhým je tzv. SOAPAction. Záhlavie SOAPAction nesie URL adresu serveru, kam bude odoslaná žiadosť HTTP SOAP komunikácie.

2.6 UDDI

Posledným komponentom technológie WS je špecifikácia pre zverejňovanie a vyhľadávanie informácií o webových službách, štandard UDDI (Universal Description, Discovery and Integration). UDDI si môžeme predstaviť ako veľký adresár, ktorý ponúka mechanizmy pre registrovanie, kategorizovanie a vyhľadávanie webových služieb. Predstavuje verejný, otvorený štandard, fungujúci pod záštitou organizácie ONASIS a je primárne určený pre obchodné využitie. Poskytovateľ WS, ktorý chce zverejniť svoju WS v UDDI, musí prirodzene službu zaregistrovať. Registrácia webovej služby je zložená z troch častí:

- **Biele stránky (angl. White pages)** – obsahujú základné údaje a kontakty o poskytovateľovi webovej slžby
- **Žlté stránky (angl. Yellow pages)** – zaradenie webových služieb na základe štandardných klasifikátorov jednotlivých obchodných a priemyselných odvetví
- **Zelené stránky (angl. Green pages)** – technické informácie o webovej službe uvoľnené poskytovateľom služby [7]

Aj samotný register UDDI je platformne nezávislou webovou službou, podlieha teda všetkým štandardom technológie webových služieb a komunikuje prostredníctvom SOAP správ.

Myšlienka vytvorenia jednotného registru webových služieb bola veľmi sľubnou, opak je však pravdou. Skutočnosť, že UDDI je verejnou službou, prináša obrovský nedostatok v jej praktickom použití. Ktokoľvek totiž môže registrovať svoju webovú službu v tomto registri bez akejkoľvek spätnej kontroly o jej pravdivosti. Z tohto dôvodu sa stal register UDDI, ako značne nespoľahlivý zdroj informácii o uložených webových službách. V praxi sa totiž ukázalo, že len dve tretiny záznamov v UDDI databáze sú platné a dôveryhodné, a zbytok záznamov je nedôveryhodných [4].

3 Vlastnosti a funkcie klienta WS

Praktickou úlohou bakalárskej práce je návrh a implementácia aplikácie klienta WS pre mobilné telefóny. V tretej kapitole sa budeme zaoberať základnými vlastnosťami a činnosťami, ktorými by mala disponovať výsledná aplikácia. Podrobnejšiu analýzu a návrh aplikácie si ukážeme v štvrtej kapitole.

Najskôr sa zamyslíme nad pojmom klient a pokúsime sa vysvetliť, čo tento pojem znamená vo svete informačných technológii. Ako bolo spomínané v úvodnej kapitole, väčšina webových aplikácií, vrátane webových služieb, je založená na architektúre klient-server. Vo svete informačných technológií sa označuje pojmom klient, aplikácia fungujúca na strane užívateľa a serverom je časť aplikácie fungujúca na strane serveru, respektíve na strane poskytovateľa. Keďže s aplikáciou klienta komunikuje priamo laický užívateľ, mala by byť maximálne užívateľsky priateľskou. To znamená, že pri návrhu aplikácie treba klásť veľký dôraz na jednoduchosť ovládania a prehľadnosť prvkov. Prvý kontakt s aplikáciou by nemal odradiť užívateľa od jej používania, ale naopak, aplikácia by mala byť užívateľovi od začiatku príjemná. Aby sme dosiahli takéhoto výsledku, musí byť aplikácia dostatočne efektívna a disponovať vhodne navrhnutými a riešenými programovými činnosťami.

3.1 Efektívna mobilná aplikácia

Prvým dôležitým faktom, ktorý si musíme uvedomiť pred návrhom a implementáciou aplikácie klienta webových služieb je fakt, že aplikácia je určená pre mobilné zariadenia. Mobilné zariadenia sú značne limitované, oproti stolovým systémom, hardvérovým vybavením, čím nám kladú isté obmedzenia pri ich implementácii. Aby sme boli schopný čeliť týmto obmedzeniam je vhodné dodržiavať pri implementácii pár nápomocných pravidiel, s ktorých pomocou môžeme dosiahnuť maximálnu efektívnosť výslednej aplikácie.

- **Vytváranie malých aplikácií.** Malé aplikácie kladú nízke nároky na pamäťový priestor mobilného zariadenia a disponujú krátkou dobou inštalácie.

- **Zníženie využitia operačnej pamäte pri behu programu.** Limitovaná je i operačná pamäť mobilného zariadenia. Z tohto dôvodu je vhodné implementovať aplikáciu pokiaľ možno tak, aby pri svojom behu využívala, čo najmenej operačnej pamäte. Pri implementácii je vhodné využívať skalárne typy a funkciu garbage collectoru.

- **Technika Lazy Instantiation.** Alokáciou len nevyhnutných objektov, dochádza k redukcii celkového a vrcholového využitia pamäti. Pri implementácii sa snažíme zohľadňovať alokáciu každého objektu.

- **Skoré uvoľňovanie zdrojov.** Skorým uvoľňovaním zdrojov uvoľňujeme operačnú pamäť mobilného zariadenia.

- **Znovu používanie objektov**. Inicializácia a deinicializácia objektov podľa potreby a využitia v aplikácii.

- **Používanie lokálnych premenných.**

- **Optimalizácia slučiek**. Optimalizáciou slučiek je možné znížiť počet cyklov iteračných príkazov zo zachovaním rovnakého výsledku.

- **Vyhýbanie sa konkatenácii**. Konkatenácia sa vyskytuje hlavne pri implementácii v progamovacom jazyku Java. Java totiž umožňuje neefektívne zlučovanie reťazcov jednoduchým sčítaním reťazcov (String + String). Efektívnejšou metódou je využitie objektu StringBuffer, ktorý sa raz skonštruuje a následne je na ňom volaná metóda append, ktorej úlohou je spojenie reťazcov.

- **Bufferovaný I/O prúd**. Bufferovaný vstup a výstup je ďaleko efektívnejšia métóda načítania dát z dátových prúdov.

- **Oddelená softvérová architektúra**. Efektívnou metódou implementácie, ktorá uľahčuje následnú úpravu aplikácie je rozdelenie aplikácie do dvoch nezávislých komponent. Je vhodné implementovať aplikáciu tak, aby GUI bolo oddelené od riadiacej logiky. V prípade nutnosti adaptovať GUI pre iný typ mobilného zariadenia je úprava kódu jednoduchšia a naopak v prípade úpravy riadiacej logiky netreba brať ohľad na GUI [8].

3.2 Klient WS

V predchádzajúcej kapitole boli popísané základné pravidlá pre efektívny návrh a implementáciu aplikácii pre mobilné zariadenia. Teraz si v krátkosti ukážeme základné činnosti, ktoré by mala obsahovať aplikácia klienta webových služieb. V praktickej časti sa potom budeme snažiť o praktické vyžitie spomínaných pravidiel a činností v našej aplikácii.

Aplikácie klientov webových služieb je možné rozdeliť do dvoch skupín. Do prvej skupiny patria aplikácie, ktoré sú vytvorené priamo pre konkrétnu webovú službu. Vnútorná architektúra a riadiaca logika takýchto aplikácii býva obyčajne veľmi jednoduchá a aplikácie často nepracujú vôbec so štandardom WSDL. Takáto aplikácia je implementovaná staticky programátorom, ktorý má k dispozícii WSDL popis služby, a samotná aplikácia už komunikuje a pracuje len so samotným štandardom SOAP. Prakticky vypadá algoritmus obsluhy statického klienta tak, že užívateľ po spustení aplikácie zadá vstupné dáta do formuláru a odošle dáta serveru. Server spracuje dáta a odošle spätnú odpoveď, ktorú aplikácia klienta príjme, spracuje a zobrazí na výslednej obrazovke.

Druhú skupinu klientov webových služieb reprezentujú aplikácie, ktoré sú určené všeobecne pre akúkoľvek webovú službu. Túto skupinu môžeme označiť ako dynamický, alebo generický klienti webových služieb. Oproti prvej skupine klientov sú aplikácie o niečo komplikovanejšie a

náročnejšie na pamäť a výkon mobilného zariadenia, alebo PC. Zložitejšia je hlavne ich riadiaca logika, nakoľko je v jej réžii syntaktická analýza WSDL dokumentu a následné generovanie správ protokolu SOAP s príslušným GUI. Praktický algoritmus obsluhy je rozšírený o výber, alebo zadanie webovej služby, následná syntaktická analýza WSDL dokumentu, ktorá prebieha na internej úrovni a vygenerovanie GUI pre zadanie vstupných dát od užívateľa. Ďalší priebeh je obdobný statickým klientom.

U generických klientov je zložitejšie i grafické užívateľské rozhranie. Jeho vstupné a výstupné formuláre SOAP komunikácie budú generované na základe popisu danej webovej služby. Aj hlavné menu bude rozšírené minimálne o jeden formulár pre vstupnú URL adresu serveru s WSDL popisom služby. Ďaleko efektívnejšie a pohodlnejšie riešenie je poskytnúť užívateľovi funkciu akejsi databázy webových služieb, do ktorej si môže užívateľ uložiť väčšie množstvo URL adries webových služieb. Na záver uvedieme v tabuľke 1 stručný prehľad základných činností statického a generického klienta webových služieb a v tabuľke 2 hlavné výhody a nevýhody jednotlivých klientov.

Statický klient Webových služieb	**Generický klient Webových služieb**
Grafické užívateľské rozhranie	**Grafické užívateľské rozhranie**
- Hlavná obrazovka s menu - Formulár pre vstupné dáta - Výstupný formulár	- Hlavná obrazovka s menu - Formulár pre zadanie URL adresy, prípadne databáza s uloženými Webovými službami - Formulár pre vstupné data - Výstupný formulár
Vnútorná logika	**Vnútorná logika**
- Komunikácia so serverom - Odosielanie SOAP požiadavku serveru - Príjem SOAP odpovede zo serveru - Získanie dát zo SOAP odpovede - Zobrazenie získaných dát vo výstupnom formuláre	- Komunikácia so serverom - Získanie WSDL dokumentu so serveru poskytovateľa WS - Syntaktická analýza WSDL dokumentu s výslednou štruktúrou Webovej služby - Generovanie vstupného a výstupného formulára - Odosielanie SOAP požiadavku serveru - Príjem SOAP odpovede zo serveru - Získanie dát zo SOAP odpovede - Zobrazenie získaných dát vo výstupnom formuláre

Tabuľka 1: Prehľad základných vlastností statického a generického klienta WS.

Statický klient Webových služieb	Generický klient Webových služieb
Výhody:	**Výhody:**
- Rýchlejší získanie informácii z webovej služby - Nízke požiadavky na pamäť a procesor mobilného zariadenia	- Nezávislosť na webovej službe - Možnosť ukladania a správy používaných webových služieb
Nevýhody:	**Nevýhody:**
- Obmedzenie využitia, často len na jednu webovú službu	- Zdĺhavejší proces získania informácii z danej webovej služby (aplikácia analyzuje WSDL) - Vyššie požiadavky na pamäť a procesor mobilného zariadenia

Tabuľka 2: Výhody a nevýhody statického a generického klienta WS.

4 Návrh implementácie

Praktickou časťou bakalárskej práce bolo vytvoriť aplikáciu klienta webových služieb s dynamickým GUI pre mobilné telefóny. V tejto kapitole sa budeme zaoberať návrhom implementácie klienta WS. Kapitola je rozdelená do dvoch podkapitol, so zámerom rozdelenia návrhu, implementácie a v konečnom dôsledku i výslednej aplikácie do dvoch softvérových architektúr. V prvej podkapitole sa budeme venovať návrhu funkčnej časti klienta a v druhej časti sa pozrieme na návrh grafického užívateľského rozhrania.

4.1 Návrh funkčnej časti klienta

Každá aplikácia, či už stolová, webová, alebo mobilná obsahuje, pre bežných užívateľov skrytú a znalosť nevyžadujúcu, vnútornú riadiacu logiku. Riadiaca logika má za úlohu vytvárať celistvú funkčnú štruktúru koncových bodov aplikácie a spracovávať logické operácie dané vlastnou implementáciou. Koncovými bodmi aplikácie môžu byť napríklad prvky GUI, alebo dáta uložené lokálne, prípadne na vzdialenom servery.
Vnútorná riadiaca logika je hlavným pilierom, na ktorom stojí chod celej aplikácie, preto je treba brať veľký dôraz pri jej návrhu a implementácii. Samozrejme svoju podstatnú úlohu a zmysel má aj grafické užívateľské rozhranie, ktoré užívateľovi umožňuje zasahovať do vnútornej logiky aplikácie z reálneho sveta. Vhodne riešeným užívateľským rozhraním sa stáva aplikácia užívateľsky priateľskou, čo je veľmi podstatná vlastnosť v IT komerčnej sfére.

Časť riadiacej logiky generického klienta je podstatne obšírnejšia a komplikovanejšia časti GUI, disponuje však menší počtom zásahov užívateľa do jej chodu. Kompletná činnosť vnútornej logiky bude pozostávať zo šiestich základných procesov, o ktorých sme sa už v krátkosti zmienili v kapitole 3.2. V tejto kapitole sa im budeme venovať podrobnejšie a ukážeme si aj ich vzájomnú interaktivitu.

- Procesy hlavného GUI a spracovanie vstupných dát zadaných užívateľom
- HTTP komunikácia
- Syntaktická analýza WSDL dokumentu
- Generovanie štruktúry Webovej služby
- Generovanie vstupného a výstupného GUI pre komunikáciu s WS
- SOAP parser

Procesy hlavného GUI – Procesy hlavného GUI reprezentujú skupinu základných procesov hlavného MIDletu, zobrazujúcich obrazovky užívateľského rozhrania s následným spracovaním vstupných a výstupných dát zadaných užívateľom. Úvodná obrazovka bude zobrazovať hlavné menu aplikácie, ktorému sa budeme venovať v kapitole návrhu GUI. Zaujímavou podskupinou procesov tejto skupiny, sú procesy s názvom Databáza webových služieb. Procesy predstavujú akúsi databázu Webových služieb, ktorá umožní užívateľovu ukladanie a správu webových služieb v pamäťovom priestore (RecordStore) mobilného zariadenia.

HTTP komunikácia – Do tejto skupiny procesov patria komunikačné procesy, založené na protokole HTTP, ktorý sme popísali v kapitole 2.2. V podstate ide o dva procesy pomocou, ktorých bude aplikácia komunikovať so serverom poskytovateľa webovej služby. Prostredníctvom prvého procesu aplikácia získa z URL adresy vybranej webovej služby WSDL dokument a druhým procesom bude už samotná SOAP komunikácia s webovou službou.

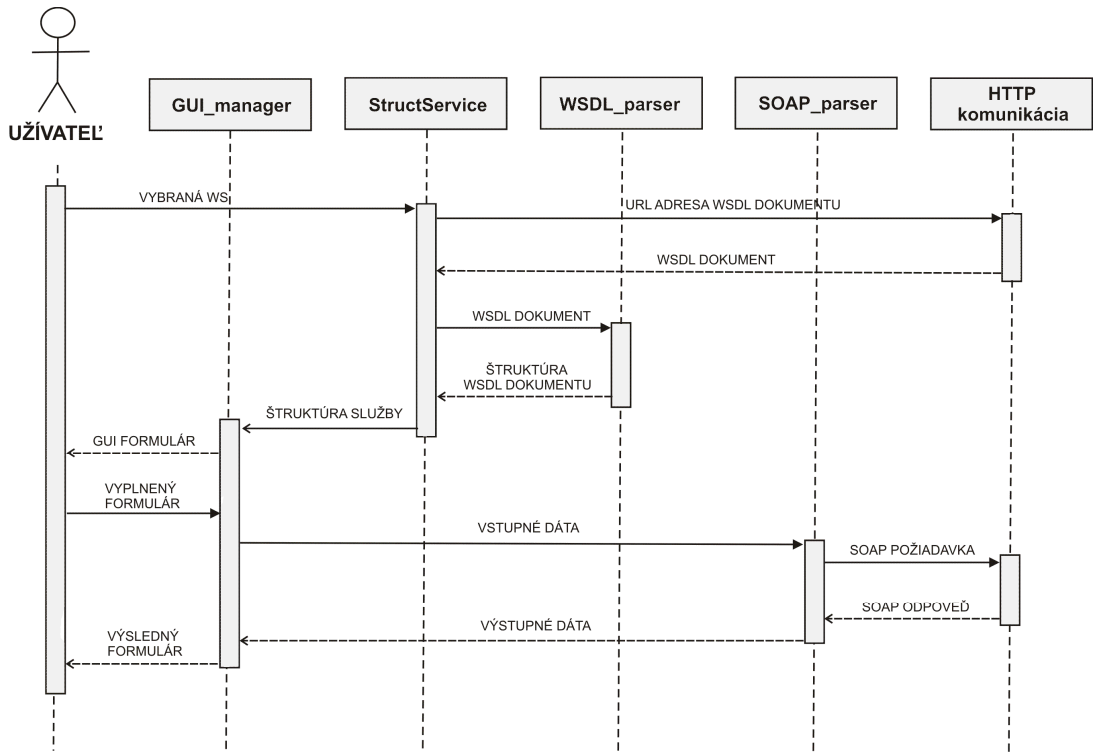
Syntaktická analýza WSDL dokumentu – Jedným z kľúčových procesov aplikácie bude syntaktická analýza WSDL dokumentu, tzv. WSDL parser (viď. kapitola 5.1.2). Úlohou procesu bude analýza WSDL popisu webových služieb s následnou reprezentáciou získaných dát do internej podoby aplikácie. Môžeme povedať, že na tomto procese bude stáť úspešnosť a rýchlosť chodu celej aplikácie.

Generovanie štruktúry webovej služby – Po získaní dát zo syntaktickej analýzy WSDL dokumentu aplikácia prostredníctvom vnútornými procesov vytvorí štruktúru webovej služby. Samotná štruktúra služby bude obsahovať zoznam operácii, ktoré služba poskytuje a štruktúru a atribúty vstupných a výstupných dát, každej operácie.

Generovanie vstupného a výstupného GUI pre komunikáciu s WS – Na základe štruktúry webovej služby, ktorú aplikácia získa z predchádzajúceho procesu analýzy WSDL, vytvorí vstupné a výstupné GUI pre SOAP komunikáciu s webovou službou.

SOAP parser – SOAP parser bude obsahovať dva základné procesy, ktoré sú úzko späté s predchádzajúcim procesom generovania vstupného a výstupného GUI. Prvý proces bude vytvárať na základe štruktúry webovej služby a dát získaných od užívateľa, SOAP požiadavku, ktorú aplikácia odošle prostredníctvom protokolu HTTP serveru webovej služby. Druhým procesom je SOAP analyzátor, ktorý získa dáta zo SOAP odpovede webovej služby. Získané dáta následne spracuje a zobrazí vo výstupnom GUI.

Na obrázku 3 je na sekvenčnom diagrame jazyka UML (Unified Modeling Language) znázornená vzájomná interaktivita popísaných procesov a štandardný chod vnútornej logiky aplikácie. Procesy sú označené podobnými názvami, ktoré sa budú objavovať ako názvy tried, prípadne balíčkov tried výslednej aplikácie. Ako prvý vstupuje do procesného sledu užívateľ výberom, alebo zadaním webovej služby spolu s URL adresou WSDL popisu. Ďalej sa vytvorí inštancia triedy Štruktúra Služby, ktorá požiada server poskytovateľa webovej služby o WSDL dokument a na základe dát získaných z analýzy WSDL dokumentu, ktoré získa prostredníctvom triedy WSDL parser, vygeneruje základnú štruktúru služby. Následne sa vytvorí nová inštancia triedy GUI manažér, ktorá príma ako parameter vytvorenú štruktúru služby a na jej základe generuje vstupné a výstupné formuláre. Užívateľovi je poskytnutý formulár pre vstupné dáta, ktoré sú následne vo forme SOAP požiadavku odoslané serveru webovej služby. Server požiadavku spracuje a vracia SOAP odpoveď, ktorej obsah zobrazí trieda GUI manažér zobrazí na konečnej obrazovke.



Obrázok 3: Sekvenčný diagram zobrazujúci vnútorný chod riadiacej logiky aplikácie pri spracovaní vybranej webovej služby a generovaní grafického užívateľského rozhrania SOAP komunikácie.

4.2 Návrh grafického užívateľského rozhrania

Grafické užívateľské rozhranie je jedným z hlavných komponentov podieľajúcich sa na finálnej spokojnosti užívateľa s aplikáciou. Z tohto dôvodu je treba klásť dostatočný dôraz pri návrhu a implementácii GUI. GUI by malo byť jednoduché, logické, prehľadné a hlavne užívateľovi príjemné.

GUI vytváranej aplikácie generického klienta webových služieb je pomerne jednoduché. Chod a ovládanie aplikácie nevyžaduje nijak komplikované základné menu, preto aj jeho návrh nebude nijak zložitý. Prakticky bude GUI aplikácie rozdelené do dvoch častí. Prvou časťou bude základné menu, ktorého hlavnou úlohou bude interpretácia možnosti voľby a správy webových služieb. Druhá časť GUI bude obsahovať formuláre generované na základe popisu webovej služby a dát získaných pri SOAP komunikácií. V prípade SOAP požiadavky bude GUI obsahovať komponenty určené pre zadanie dát od užívateľa a pri následnej SOAP odpovedi serveru dôjde k zobrazeniu získaných dát na obrazovke mobilného zariadenia.

4.2.1 Základné menu GUI

Základom GUI každej mobilnej aplikácie je úvodná obrazovka, ktorú uvidí užívateľ, ako prvú po každom spustení aplikácie. Prakticky sa dnes využívajú dva typy úvodných obrazoviek. Prvým

typom je takzvaný „Splash screen“, ktorý najčastejšie obsahuje informácie, ako názov, verziu, copyright výrobcu, kontakt na výrobcu, prípadne vlastné logo aplikácie. V druhom prípade tvorí úvodnú obrazovku ponuka hlavného menu, reprezentovaná celoobrazovkovým zoznamom položiek s možnosťou voľby. Nami vytváraný klient webových služieb bude zobrazovať, po jeho spustení, druhý typ úvodnej obrazovky s ponukou základného menu s piatimi položkami. Jednotlivé položky menu budú odkazovať na sekundárne obrazovky základného GUI aplikácie. Vypíšeme a v krátkosti spomenieme názvy a význam jednotlivých položiek menu. Na obrázku 4 je potom vidieť stromovú štruktúru základných obrazoviek vo vzťahu z položkami základného menu.

1. Položka **Spusť WS** – Prvá položka menu reprezentuje štart hlavnej časti vnútornej logiky aplikácie. Po jej výbere bude užívateľovi zobrazená sekundárna obrazovka s ponukou webových služieb uložených v pamäťovom priestore mobilného zariadenia. Po zvolení žiadanej webovej služby bude spustený proces spracovania webovej služby (viď Obrázok 3).

2. Položka **Databáza WS** - Po vybraní položky Databáza WS z ponuky menu sa užívateľovi zobrazí, rovnako ako v prvom prípade, zoznam webových služieb uložených v pamäťovom priestore mobilného zariadenia. Avšak na tejto obrazovke má užívateľ možnosť správy uložených webových služieb. V menu pod jedným z funkčných tlačidiel má k dispozícii ponuku s voľbami: Nová WS, Detail WS, Upraviť WS, Vymazať WS. Úloha a význam jednotlivých položiek sú celkom jasné už z ich názvu, preto sa nimi nebudeme ďalej zaoberať.

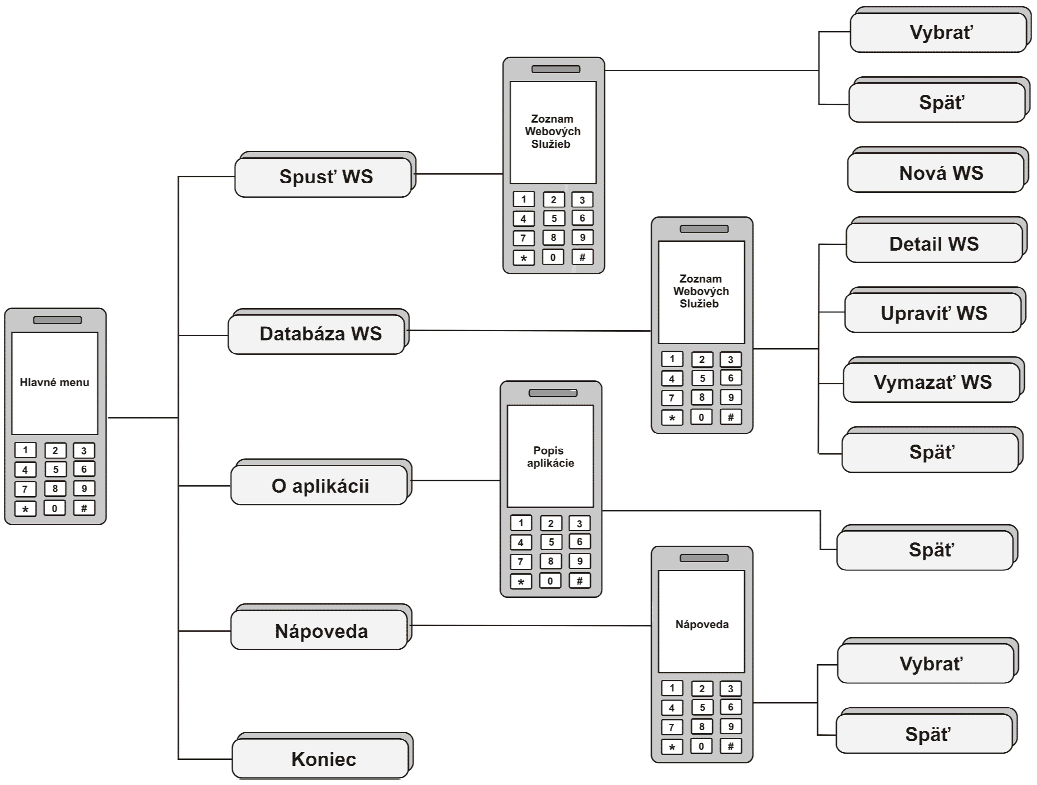
3. Položka **O aplikácii** – Táto položka menu nepotrebuje podrobné vysvetlenie. Po jej zvolení sa užívateľovi zobrazí obrazovka so základnými informáciami o aplikácii: autor, verzia, dátum, krátky popis aplikácie.

4. Položka **Pomoc** – Pod položkou Pomoc nájde užívateľ základné rady pre prácu s aplikáciou.

5. Položka **Koniec** – Poslednou položkou menu je možnosť ukončenia chodu aplikácie.

4.2.2 GUI SOAP komunikácie

Druhú časť grafického užívateľského rozhrania tvoria obrazovky, zložené z väčšej časti z formulárov, generované na základe WSDL popisu vybranej webovej služby. Jednotlivé obrazovky budú slúžiť jednak pre zadávanie údajov určených pre SOAP požiadavky a jednak pre prezentovanie SOAP odpovedí zo serveru webových služieb. Každá obrazovka bude obsahovať pod jedným z funkčných tlačidiel menu s príslušnou ponukou volieb. Položky menu jednotlivých obrazoviek budú až na pár výnimiek rovnaké a budú umožňovať odoslanie SOAP správy, spätný presun na predchádzajúce obrazovky v rámci SOAP komunikácie a návrat do hlavného menu.



Obrázok 4: Návrh štruktúry základného menu. Oválne obrázky zobrazujú položky menu na jednotlivých obrazovkách.

4.2.3 Typický chod aplikácie

V tejto podkapitole si ukážeme v jednotlivých bodoch, ako bude vypadať typický algoritmus práce s aplikáciou klienta webových služieb z pohľadu užívateľa. V jednotlivých bodoch si v krátkosti spomenieme i pridruženú vnútornú logiku, ktorá je spojená s daným vstupom užívateľa do chodu aplikácie. Podrobnejšie sú všetky funkcie popísané v užívateľskej príručke, ktorá je prílohou bakalárskej práce.

1. **Spustenie aplikácie** – Po spustení aplikácie sa dostaneme do hlavného menu.

2. **Vloženie novej webovej služby** – Voľbou položky Databáza WS z hlavného menu sa dostaneme na obrazovku, kde je možné pomocou položiek menu funkčného tlačidla vkladať, upravovať a vymazávať webové služby uložené v pamäťovom priestore mobilného zariadenia. Z menu vyberieme možnosť Nová WS. Zobrazí sa nám formulár, kde vyplníme polia s názov a URL adresa WSDL dokumentu webovej služby. Po uložení sa voľbou Späť vrátime do hlavného menu.

3. **Výber webovej služby** – V hlavnom menu vyberieme položku Spusti WS, ktorá nás presunie na obrazovku so zoznamom webových služieb. Zo zoznamu vyberieme požadovanú webovú službu a potvrdíme stredným tlačidlom. Následne začne prebiehať časť vnútornej logiky

zaoberajúca sa spracovaním WSDL dokumentu a SOAP komunikáciou. Aplikácia sa pokúsi načítať WSDL popis služby so zadanej URL adresy, následne dokument zanalyzuje a vytvorí štruktúru webovej služby. Na základe štruktúry webovej služby sa generuje prvotná obrazovka GUI SOAP komunikácie, ktorá zobrazí zoznam operácií poskytovaných webovou službou.

4. **Výber operácie** – Zo zoznamu si vyberieme žiadanú operáciu, ktorú daná webová služba poskytuje a na základe štruktúry služby nám opäť vnútorná logika vygeneruje obrazovku z formulárom pre zadanie vstupných dát pre SOAP požiadavku. Po vyplnení formulárových polí vyberieme z menu možnosť Odoslať, čím za pomoci vnútornej logiky odošleme SOAP požiadavku serveru webovej služby. Server spracuje požiadavku a vracia SOAP odpoveď.

5. **Zobrazenie výsledných dát** – Po získaní SOAP odpovede aplikácia za pomoci štruktúry webovej služby zanalyzuje SOAP správu a zobrazí získané dáta na konečnej obrazovke.

6. **Ukončenie aplikácie** – Návratom do hlavného menu a voľbou Koniec, prípadne priamou voľbou položky Koniec z menu výslednej obrazovky, ukončíme aplikáciu.

5 Implementácia

V piatej kapitole sa budeme venovať samotnej implementácii klienta WS. Kapitola je rozdelená do dvoch podkapitol. V prvej podkapitole spomenieme technológie, ktoré boli použité pri implementácii a v druhej podkapitole si potom popíšeme samotnú implementáciu jednotlivých balíčkov, ktoré reprezentujú procesy návrhu s kapitoly 4.

5.1 Použité technológie

Praktická implementácia aplikácie Klienta webových služieb bola realizovaná v programovacom jazyku Java, konkrétne v platforme Java ME (angl. Java Micro Edition). K implementácii sme využili vývojové prostredie NetBeans verzie 6.7.1, nakoľko disponuje s rozsiahlou užívateľskou základňou a je priamym produktom firmy Sun Microsystems.

Druhou technológiou, ktorá bola použitá pri implementácii je tzv. XML parser, ktorého úlohou je syntaktická analýza dokumentov vo formáte XML s následnou reprezentáciou získaných dát do internej podoby aplikácie. XML parser je nevyhnutnou súčasťou aplikácie klienta WS, nakoľko je jazyk XML základným pilierom webových služieb.

5.1.1 Java ME

Java je objektovo orientovaný programovací jazyk, založený na princípoch C a C++, ktorý bol oficiálne predstavený v roku 1995 firmou Sun. Silnou vlastnosťou programovacieho jazyka, ktorej zásluhou je jeho veľká obľúbenosť a použiteľnosť, je platformná nezávislosť. To znamená, že aplikácia vytvorená v programovacom jazyku Java je spustiteľná a použiteľná bez obmedzení na rôznych operačných systémoch: Windows, Linux apod. Práve nezávislosť Javy na platforme vzbudila veľký záujem o tento programovací jazyk, v ktorom bol vidieť obrovský potenciál využitia. Avšak vývojári vytvárajúci aplikácie pre komerčných koncových užívateľov, boli v dobe vzniku Javy zvyknutý na bohatšie užívateľské rozhranie, ktoré im ponúkali konkurenčné jazyky. Firma Sun tak bola nútená rozšíriť rozsah Java platformy a prichádza s platformou Java 2, ktorá bola tesne pred jej oficiálnym uvedením rozdelená na niekoľko častí. Toto rozdelenie vo svojej podstate zotrvalo až dodnes.

- **Java SE (Java Standard Edition)** – Platforma so štandardnou funkcionalitou, ktorá je považovaná za minimálnu podporu vyžadovanú pre ľubovoľné prostredie Java.

- **Java EE (Java Enterprise Edition)** – Platforma určená pre využitie Javy pri vývoji aplikácií na podnikovej úrovni a v prostredí aplikačných serverov. J2EE integruje nové technológie, ako napríklad servlety, Enterprise JavaBeans, JSP apod.

- **Java ME (Java Micro Edition)** – Platforma určená pre vývoj aplikácií pre mobilné zariadenia s obmedzeným množstvom voľnej pamäte a nízkym výkonom procesoru.

Java ME je platforma určená pre vývoj aplikácií pre mobilné zariadenia. Jej špecifikácia je, na rozdiel od platforiem Java SE a Java EE, rozdelená na viac častí podľa charakteristík a schopností mobilných zariadení – tzv. konfigurácií. Svet mobilných zariadení totiž obsahuje ďaleko väčšiu škálu zariadení, ktorých schopnosti sú naozaj veľmi rozmanité a odlišné. Z tohto dôvodu nie je prakticky možné, alebo len veľmi ťažko, vytvoriť jediný softvérový produkt pre všetky mobilné zariadenia.

Jednotlivé konfigurácie sú špecifikácie definujúce softvérové prostredie pre určitú škálu zariadení, respektíve reprezentujú minimálnu platformu pre dané cieľové zariadenie. V súčasnosti sú v Java ME definované dve konfigurácie:

- **CLDC (Connected Limited Device Configuration)** – CLDC je konfigurácia určená pre nízkoúrovňové zariadenia, či už mobilné telefóny, pagery, alebo PDA vybavené s voľnou pamäťou veľkosti približne 512 kB.

- **CDC (Connected Device Configuration)** – Konfigurácia určená pre zariadenia, ktoré sa nachádzajú medzi CLDC a úplnými stolovými systémami. CDC zariadenia disponujú voľnou pamäťou väčšou ako 2MB a výkonnejšími procesormi, preto môžu podporovať širšie softvérové prostredie Javy. Typické CDC zariadenia sú PDA, smartphone apod.

Nami vytváraná aplikácia je určená pre mobilné telefóny, preto budeme implementovať v konfigurácii CLDC - 1.1 platformy Java ME.

Nadstavbou CLDC, určenou predovšetkým pre mobilné telefóny, je profil MIDP. MIDP pridáva k CLDC knižnice určené pre prácu so sieťou, užívateľske rozhranie a lokálne úložisko, vo verzii MIDP 2.0 pridáva autentizačné API, herné API atď. V ďalšom texte si v krátkosti ukážeme prvky GUI, ktoré nám poskytuje MIDP pri implementácii mobilných aplikácii. Model užívateľského rozhrania pre MIDP zariadenia je v skutku modelom veľmi jednoduchým a jeho kompletná knižnica je implementovaná v balíčku `javax.microedition.lcdui`. Základným prvkom modelu je trieda `Display`, ktorá reprezentuje logickú obrazovku mobilného zariadenia. Ďalšou triedou, podieľajúcou sa na zobrazení GUI prvkov na obrazovke mobilného zariadenia je abstraktná trieda `Displayable`. Jej úlohou je pripojenie častí GUI, prípadne vykreslením grafických prvkov, k oknu triedy `Dislay`, ktoré je na najvyššej úrovni. Od triedy `Displayable` je odvodená skupina tried umožňujúca vytváranie základných užívateľských rozhraní. Z triedy `Displayable` sú priamo odvodené dve abstraktné triedy:

- **`Canvas`** – Trieda obsahuje základ nízkoúrovňových API pre grafické užívateľské rozhrania. Trieda prostredníctvom metódy `paint()` umožňuje priame kreslenie na obrazovku.

- **`Screen`** – Abstraktná trieda, od ktorej sa odvodzuje vrchol hierarchie okien vysokoúrovňového API. Trieda `Screen` na rozdiel od triedy `Canvas` neumožňuje priame kreslenie na obrazovku, ale naopak poskytuje vývojárom sadu tried umožňujúcich pohodlnejšie vytváranie GUI formulárového typ.

Pri implementácii aplikácie klienta WS budeme používať prvky vysokoúrovňového GUI, preto popis triedy `Canvas` vynecháme a prejdeme priamo k popisu triedy `Screen`. Potomkami triedy `Screen` sú triedy `Alert`, `Form`, `List` a `TextBox`, ktoré sa vužívajú spolu so sadou objektov `Command` k vytváraniu formulárového API mobilných aplikácii. Každá zo štyroch spomínaných tried má svoju funkciu a typický vzhľad, čím charakterizuje svoje využitie v aplikáciách. Hlavnými GUI prvkami, ktoré budeme využívať v našej aplikácii klienta WS budú komponenty `List` a `Form`. Objekty typu `List` použijeme pre zobrazenie menu formou zoznamu s možnosťou voľby a objekty typu `Form` použijeme pre prezentáciu a zadávanie parametrov. Triedy `List` i `Form` sú celoobrazovkové prvky, kde `List` zobrazuje zoznam a `Form` zobrazuje formulár s položkami, ktoré môžu byť rôzneho typu:

- **`StringItem`** – možnosť umiestiť do textového poľa GUI reťazec
- **`TextField`** – jednoriadkové vstupné pole s nadpisom
- **`DateField`** – umožňuje zadanie do textového poľa údaje dátového typu
- **`ChoiceGroup`** – komponenta poskytujúca sadu volieb, ktoré sa môžu, ale nemusia vzájomne vylučovať (zaškrtávacie políčka, alebo prepínač)
- **`ImageItem`** – umožňuje umiestniť obrázky do formuláru
- **`Gauge`** – umožňuje zobrazenie pokroku prebiehajúcej operácie, alebo k zaisteniu výberu hodnôt zo súvislej rady hodnôt [9].

5.1.2 XML parser

Ako už vieme, základom technológie webových služieb je jazyk XML, ktorý sme si v krátkosti popísali v druhej kapitole. Z toho nám je hneď jasné, že sa pri implementácii generického klienta webových služieb, nevyhneme práci s dátami vo formáte XML. V praxi sa označuje spracovanie dát XML formátu názvom parsovanie a jeho výsledkom je prevod XML dokumentu do internej dátovej reprezentácie. Jednou z možností, ako parsovať XML dokumentu, je využitie už naimplementovaných voľne dostupných knižníc, tzv. XML parserov. V súčasnosti existuje niekoľko XML parserov určených pre jazyk Java, či už pre platformu Java SE, Java EE, alebo pre platformu Java ME. Nakoľko platforma Java ME nedisponuje všetkými triedami a rozhraniami štandardnej platformy Java, i samotné XML parsery tejto platformy neobsahujú kompletný balík funkcií, s ktorými sa stretávame u bežných parserov. Avšak hlavnou funkciou, s ktorou sa určite stretneme u akéhokoľvek XML parseru, je syntaktická analýza XML dokumentu s jeho následnou reprezentáciou, či už do internej stromovej štruktúry, alebo do série udalostí. Práve na základe výslednej reprezentácie získaných dát sa XML parsery rozdeľujú na tri základné typy [10]:

- **Typ model** – Tento typ parseru načíta celý XML dokument a následne na základe načítaných dát vytvorí v pamäti stromovú štruktúru objektov, ktoré reprezentujú pôvodný dokument. Takmer všetky parsery typu model podporujú stromovú štruktúru DOM (angl. Document Object Model), ktorá je špecifikáciou programového rozhrania W3C konzorcia a je použiteľná v mnohých prostrediach a aplikáciách. DOM definuje logickú štruktúru dokumentu a spôsob prístupu a manipulácie s dokumentom. Typickým reprezentantom tohto typu je NanoXML/Lite parser určený pre platformu Java ME

- **Typ push** – XML parser typu push využíva pri svojej činnosti API založené na udalostiach, tzv. SAX (angl. Simple API for XML). Parser pri spracovaní dokumentu

negeneruje stromovú štruktúru objektov, ale sériu udalostí, na ktoré aplikácia patričným spôsobom reaguje. V tomto prípade nie je nutné udržovať celý dokument v pamäti, čím zvýšime použiteľnosť aplikácie na väčšom množstve nízkoúrovňových zariadení.

- **Typ pull** – Poslednou skupinou sú parsery typu pull. Tento typ parseru číta XML dokument po častiach, ktoré sú postupne spracovávané aplikáciou. Parsery typu pull využívajú pre svoj chod XML pull API a ich typickým zástupcom určeným pre platformu Java ME je kXML parser, ktorý si v krátkosti popíšeme.

kXML – kXML je parser vyvinutý priamo pre použitie v MIDP aplikáciách a z tohto dôvodu je ukrátený o niektoré funkcie bežných XML parserov. V kXML by sme napríklad zbytočne hľadali podporu validácie dokumentov pomocou DTD, alebo XML schémy. Parser patrí do skupiny typu pull, vyžaduje teda menšie pamäťové nároky na úkor väčšieho rozsahu pamäťového priestoru vo výslednom JAR súbore a v konečnom dôsledku i v samotnom mobilnom zariadení. Dnešné mobilné zariadenia sú však vybavené dostatočným množstvom pamäťového priestoru, s podporou ďalšieho rozšírenia prostredníctvom pamäťových kariet. Medzi základné vlastnosti kXML parseru patria [9]:

- podpora menných priestorov (angl namespaces)
- parsrovanie vnoreného/modulového obsahu
- malá pamäťová náročnosť
- voliteľné: podpora kDOM
- voliteľné: podpora WBXML/WML

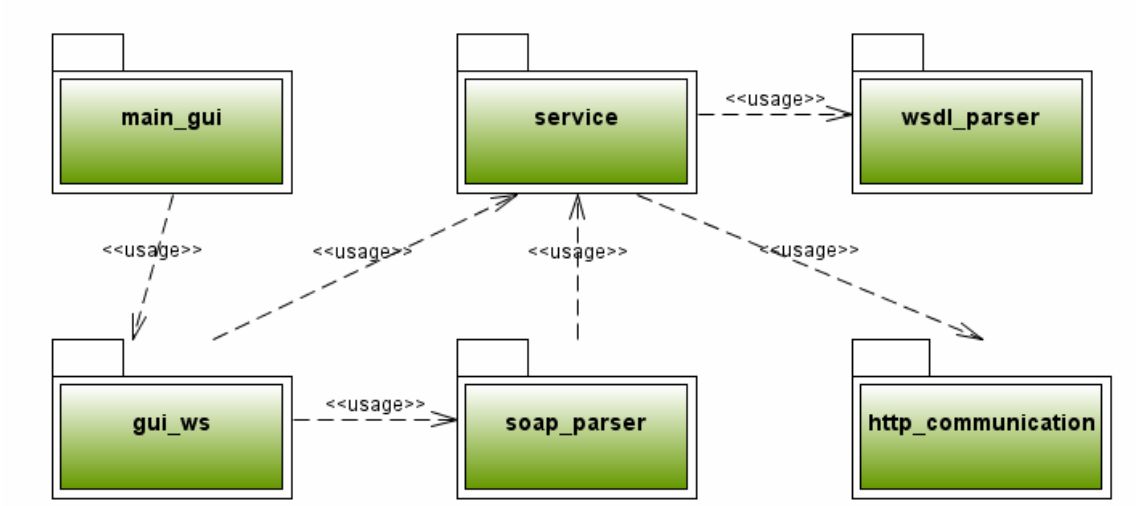
Pri praktickej implementácii aplikácie, ktorá bude pracovať s dátami v XML formáte, máme dve možnosti. Jednou z možností je vytvoriť si vlastný XML parser, alebo použiť niektorú z už naimplementovaných voľne dostupných knižníc. V našom prípade, kedy implementujeme aplikáciu určenú pre mobilné telefóny, sa javí ako najvhodnejším riešením využiť knižnicu kXML parseru. Napriek tomu bolo našim konečným rozhodnutím vytvoriť si vlastný XML parser. Za týmto rozhodnutím stál jednak fakt, že XML dokumenty prítomné v jednoduchých webových službách nie sú nijak zložité a náročné na spracovanie a ich štruktúra je striktne daná štandardom WSDL a protokolom SOAP. Druhým faktom je, že pri implementácii vlastného parseru sme získali prehľad o základných elementoch WSDL a SOAP dokumentov a mali plnú kontrolu nad voľbou potrebných elementov. Na druhú stranu, implementácia vlastného XML parseru bola časovo náročnejšia a pracnejšia.

5.2 Implementácia balíčkov

V kapitole 4.1 Návrh funkčnej časti klienta sme navrhli 6 základných procesov, respektíve skupín procesov, ktoré budú spoločne vytvárať výslednú aplikáciu klienta webových služieb. V tejto kapitole sa pokúsime jednoducho, prehľadne a zrozumiteľne popísať praktickú implementáciu balíčkov, ktoré budú prevádzať procesy návrhu.

Pre vyššiu prehľadnosť, orientáciu v zdrojových kódoch aplikácie a možno i ďalšie využitie, je aplikácia rozdelená do 6 samostatných balíčkov, ktorých obsahom sú samotné triedy

programovacieho jazyka Java. Na obrázku 5 je zobrazená štruktúra a vzájomná interaktivita balíčkov, ktoré vytvárajú výslednú aplikáciu klienta WS. Podrobnejšie diagramy tried sú uvedené v prílohách bakalárskej práce, prípadne pri popise jednotlivých balíčkov v nasledujúcej časti textu.



Obrázok 5: Štruktúra balíčkov, ktoré spoločne vytvárajú výslednú aplikáciu klienta WS.

5.2.1 Balíček main_gui

Balíček obsahuje triedy reprezentujúce základné menu s položkami umožňujúcimi orientáciu v aplikácii. Hlavné menu je vytvorené pomocou komponentu List nachádzajúceho sa v balíčku: `javax.microedition.lcdui.List`. Prevádzkový režim komponentu `List` je nastavený na hodnotu `IMPLICIT`, čím sme získali zoznam s obmedzením výberu len jednej položky so vzhľadom štandardného zoznamu.

Funkčnosť menu je riešená prostredníctvom poslucháča `CommandListener` komponentu `List`, ktorý je upozornený na zmenu výberu v okamžiku výberu a stlačenia tlačidla Select mobilného telefónu. Po zavolaní metódy `commandAction()` poslucháča `CommandListener` je metóde predaný ako parameter príkaz `Command`, ktorý bol aktivovaný a trieda `Displayable`, ku ktorej bol daný `Command` pripojený. Na obrázku 6 je časť kódu s riešením asociácie poslucháča `CommandListener` s komponentom List pomocou metódy commandAction() a detekcia zmeny komponentu List v metóde commandAction(). Výberom jednotlivých položiek menu sa dostaneme na sekundárne obrazovky, ktoré sú rovnako, ako hlavné menu, riešené celoobrazovkovým komponentom `List`. Po výbere prvých dvoch položiek menu sa nám zobrazí na novej obrazovke zoznam uložených webových služieb v mobilnom telefóne. Prvá položka v menu umožňuje výber a spustenie webovej služby a pod druhou položkou máme možnosť správy uložených webových služieb. Webové služby sú uložené v trvalom úložnom priestore mobilného zariadenia, ktorého skladovací mechanizmus sa môže líšiť u jednotlivých zariadení. Nemenné však zostáva programovacie rozhranie, ktoré poskytuje Java ME v rozhraní `RecordStore` implementovanom v balíčku `javax.microedition.rms`. V našej aplikácii je obsluha rozhrania `RecordStore` implementovaná v triede `RecordStoreWS.java` balíčku `rus107.main_gui`.

```
List menu = new List ("Klient WS", List.IMPLICIT);
menu.append("Spusť WS", run_ws);
menu.append("Databáza WS", store_ws);
menu.append("O aplikácii", oalpikacii);
menu.append("Nápoveda", help);
menu.append("Koniec", close);
menu.setCommandListener(this);

public void commandAction(Command c, Displayable d) {
   try
   {
      if (c == List.SELECT_COMMAND)
      {
         List list = (List)d;
         int index = list.getSelectedIndex();

         switch(index) {
              case 0: SpustiWS();
                       break;
              case 1: DatabaseWS();
                       break;
              case 2: Oaplikacii();
                       break;
              case 3: Napoveda();
                       break;
              case 4: destroyApp(true);
                       break;
              default: destroyApp(true);
          }
      }
   }
   catch(Exception e) { System.err.println(e.toString()); }
}
```

Obrázok 6: Asociácie poslucháča `CommandListener` s komponentou `List` pomocou metódy `commandAction()` a detekcia zmeny komponentu `List` v metóde `commandAction()`.

Trieda obsahuje 6 metód, ktoré umožňujú ukladanie nových a manipuláciu už s uloženými webovými službami.

```
public void Open_ulozisko()
public void Zavri_ulozisko()
public void Read_poleZaznamov()
public void Vlozit_WS(ZaznamWS zaznam)
public boolean Upravit_WS(ZaznamWS zaznam)
public boolean Vymazat_WS(int id)
```

Prvé tri metódy slúžia pre otvorenie a zatvorenie spojenia s trvalým pamäťovým priestorom a načítanie záznamov webových služieb do voľnej pamäte mobilného telefónu. Ďalšie tri metódy umožňujú vkladanie, vymazávania a upravovanie jednotlivých záznamov WS. Mechanizmus

databázy WS pracuje s objektmi typu ZáznamWS, ktoré vlastnia tri premenné: idWS, menoWS a urlWS. Na obrázku 7 je vidieť úsek kódu práce s rozhraním RecordStore.

```
RecordSore storeWS;

public void Open_ulozisko()
{
   try {
 storeWS = RecordStore.openRecordStore(meno_uloziska, true);
   } catch(Exception e) { System.err.println(e.toString());}
}

public void Zavri_ulozisko()
{
   try {
       if(storeWS != null)
             storeWS.closeRecordStore();
   } catch(Exception e) { System.err.println(e.toString());}
}

public void Vlozit_WS(ZaznamWS zaznam)
{
   try {
      int storeID = storeWS.getNextRecordID();
      ByteArrayOutputStream baos = new ByteArrayOutputStream();
      DataOutputStream os = new DataOutputStream(baos);

      os.writeInt(storeID);
      os.writeUTF(zaznam.getMenoWS());
      os.writeUTF(zaznam.getUrlWS());
      os.close();

      byte[] data = baos.toByteArray();
      int id = storeWS.addRecord(data, 0, data.length);
      Read_poleZaznamov();
   }
   catch(Exception e) { System.err.println(e.toString());}
}
```

Obrázok 7: Blok kódu zobrazujúci základnú prácu s rozhraním RecordStore, ako je otvorenie spojeniam, zatvorenie spojenia a vloženie záznamu webovej služby.

5.2.2 Balíček http_communication

Komunikáciu klienta so serverom poskytovateľa WS sme riešili pomocou protokolu HTTP, ktorého podporu nájdeme v každom MIDP zariadení. Java ME poskytuje komplexnú podporu sieťovej komunikácie prostredníctvom rámcového systému GCF (angl. Generic Connection Framework) konfigurácie CLDC, ktorý je implementovaný v balíčku `javax.microedition.io`. Kľúčovými prvkami tohto balíčku sú trieda `Connector` a rozhranie `Connection`, ktorého potomkom je rozhranie `HttpConnection` určené pre vytvorenie HTTP spojenia. Inštanciu `HttpConnection` získame zavolaním métody `Connector open()` s parametrom URL adresy serveru. Tým sa naviaže spojenie a ďalej už nasleduje samotná komunikácia a prenos dát. Volaním medódy `setRequestMethod()` nastavíme typu metódy žiadosti na `POST` a importujeme

voliteľné hlavičky HTTP. Následne sú dáta zapísané do výstupného prúdu volaním metódy `openOutputStream()`, čím sú odoslané serveru. Pre príjem nastavíme typ metódy žiadosti na `GET` a prijaté data načítame zo vstupného prúdu `InputStream()` pomocou metódy `openInputStream()`. Iným prípadom je situácia, kedy neodosielame serveru žiadne dáta, ale dáta zo serveru len prijímame. V našej aplikácii ide napríklad o príjem WSDL dokumentu. V tomto prípade nastavíme typ metódy žiadosti `GET` ihneď na začiatku metódy. Následné načítanie dát je obdobné prvému prípadu.

```
HttpConnection con = null;
InputStream is = null;
OutputStream os = null;
if(!targetNamespace.endsWith("/"))
            targetNamespace += "/";

try {
   con = (HttpConnection) Connector.open(url);
   con.setRequestMethod(HttpConnection.POST);
   con.setRequestProperty("Content-Type","text/xml;
                                          charset=utf-8");
   con.setRequestProperty("SOAPAction","\"" + targetNamespace +
                            operation + "\"" );

   os = con.openOutputStream();
   os.write(message.getBytes());

   con.setRequestMethod(HttpConnection.GET);
   is = con.openInputStream();
   String ct = con.getHeaderField("content-type");

   InputStreamReader isr = new InputStreamReader
               (is,ct.substring(ct.lastIndexOf('=')+1));
            String s = null;

// tu sa nachadza konverzia dát na textový reťazec String

   if(is != null) { is.close(); }
   if(con != null) { con.close(); }
   return s;
}
catch (IOException e) {
        e.printStackTrace();
        return null; }
```

Obrázok 8: Časť kódu metódy `Connect` triedy `ConnectToServer.java`, demonštrujúci riešenie SOAP komunikácie klienta a serveru poskytovateľa webových služieb.

V tomto úvode sme popísali praktické riešenie HTTP komunikácie, ako sme ho riešili v našej aplikácii. Konkrétne nám k HTTP komunikácii slúži trieda `ConnectToSerever.java` balíčku `rus107.klientWS`, ktorá obsahuje dve metódy:

`public static String ReadWSDL(String url)`
Metóda príma jeden parameter s URL adresou serveru a vracia získané dáta skonvertované do textového reťazca. Úlohou metódy je získať WSDL dokument zo serveru poskytovateľa WS na základe zadanej URL adresy. Metóda obsahuje len časť komunikácie GET, nakoľko metóda neodosiela, ale len prijíma dáta.

`public static String Connect (String url, String message, String targetNamespace, String operation)`
Ďaleko zaujímavejšia je metóda Connect, ktorej úlohou je už samotná SOAP komunikácia klienta so serverom WS. Metóda príma 4 parametre: URL adresu serveru, SOAP správu, ktorá má byť serveru odoslaná a targetNamespace spolu s názvom operácie, ktoré budú vložené ako hlavička HTTP. Metóda po vytvorení spojenia, nastaví metódu žiadosti a vloží nevyhnutné údaje hlavičky `Content-Type` a `SOAPAction`. Ďalej nasleduje odoslanie správy serveru WS, príjem odpovede metódou žiadosti GET a konverzia odpovede do formátu textového reťazca. Textový reťazec s odpoveďou serveru je aj návratovou hodnotou metódy. Na obrázku 8 je ukážková časť kódu práve z metódy Connect.

5.2.3 Balíček wsdl_parser

Proces syntaktickej analýzy zabezpečuje aplikácia prostredníctvom XML parseru, ktorého funkčnosť je popísaná v podkapitole 5.1.2. V našej aplikácii sme implementovali vlastný XML parser, ktorý bude pracovať priamo s elementmi štandardu WSDL. A práve na základe toho, že pracuje priamo s elementmi štandardu WSDL, označili sme ho v našom projekte, ako WSDL parser.

Nami implementovaný WSDL parser je v podstate jednoduchý XML parser čiastočne založený na type model, z čoho vyplýva, že v pamäti vytvára stromovú štruktúru objektov WSDL dokumentu. Pojem čiastočne sme použili z toho dôvodu, že parser nevytvára v pamäti úplnú štruktúru WSDL dokumentu, ale len čiastočnú. Nakoľko sme si mohli analýzu nami vytváraného parseru prispôsobiť podľa našich potrieb, parser vytvára v pamäti štruktúru zloženú len z nevyhnutných elementov WSDL popisu služby. Tým sme do istej miery znížili zaťaženie procesoru a spotrebu voľnej pamäte mobilného telefónu.

Knižnica WSDL parseru sa nachádza v balíčku `rus107.wsdl_parser`, ktorý obsahuje 18 tried. Z toho 11 tried zahŕňa analýzu elementov WSDL, 5 tried analýzu XML schémy k definícii dátových typov a dve triedy sú spoločné pre obidve skupiny. Základnou, alebo koreňovou triedou štruktúry je trieda `WSDL_parser.java`, ktorá uchováva základné objekty WSDL dokumentu ako: types, messages, portType, binding a service (viď. kapitolu 2.4). Každý zo základných elementov pokračuje v analýze a ukladá vnorené elementy na základe štandardu WSDL. Výsledkom je jednopriechodový WSDL parser. K ukladaniu a manipulácii objektov rovnakého typu využíva

parser dátovú štruktúru `Vector` a jednotlivé atribúty elementov vkladá do dátovej štruktúry `HashTable`, kde kľúčom je názov atribútu a hodnotou je jeho hodnota v elemente.

5.2.4 Balíček service

Ďalším procesom návrhu, ktorý je treba prakticky naimplementovať, je proces generovania štruktúry webovej služby, ktorý sme implementovali v balíčku `rus107.Service`. Štruktúrou webovej služby myslíme zjednotenie nevyhnutných informácii o webovej službe získaných z analýzy WSDL dokumentu, ktoré budeme potrebovať pri generovanií GUI a SOAP komunikácii.

Štruktúra služby nesie informácie o danej webovej službe zoskupené do stromovej štruktúry (Obrázok 9). Koreňom štruktúry je samotná služba, jej potomkami sú jednotlivé operácie, ktoré webová služba poskytuje. Každá z operácií ďalej obsahuje vstupný (angl input) a výstupný (angl. output) objekt, ktorý obsahuje pole objektov reprezentujúcich základné elementy. Objekty nesú informácie o názve a dátovom type základných elementov a určujú aplikácii koľko bude prítomných parametrov v SOAP komunikácii s webovou službou. Štruktúra služby pozostáva zo štyroch tried, ktoré nájdeme už v spomínanom balíčku `rus.107.Service`:

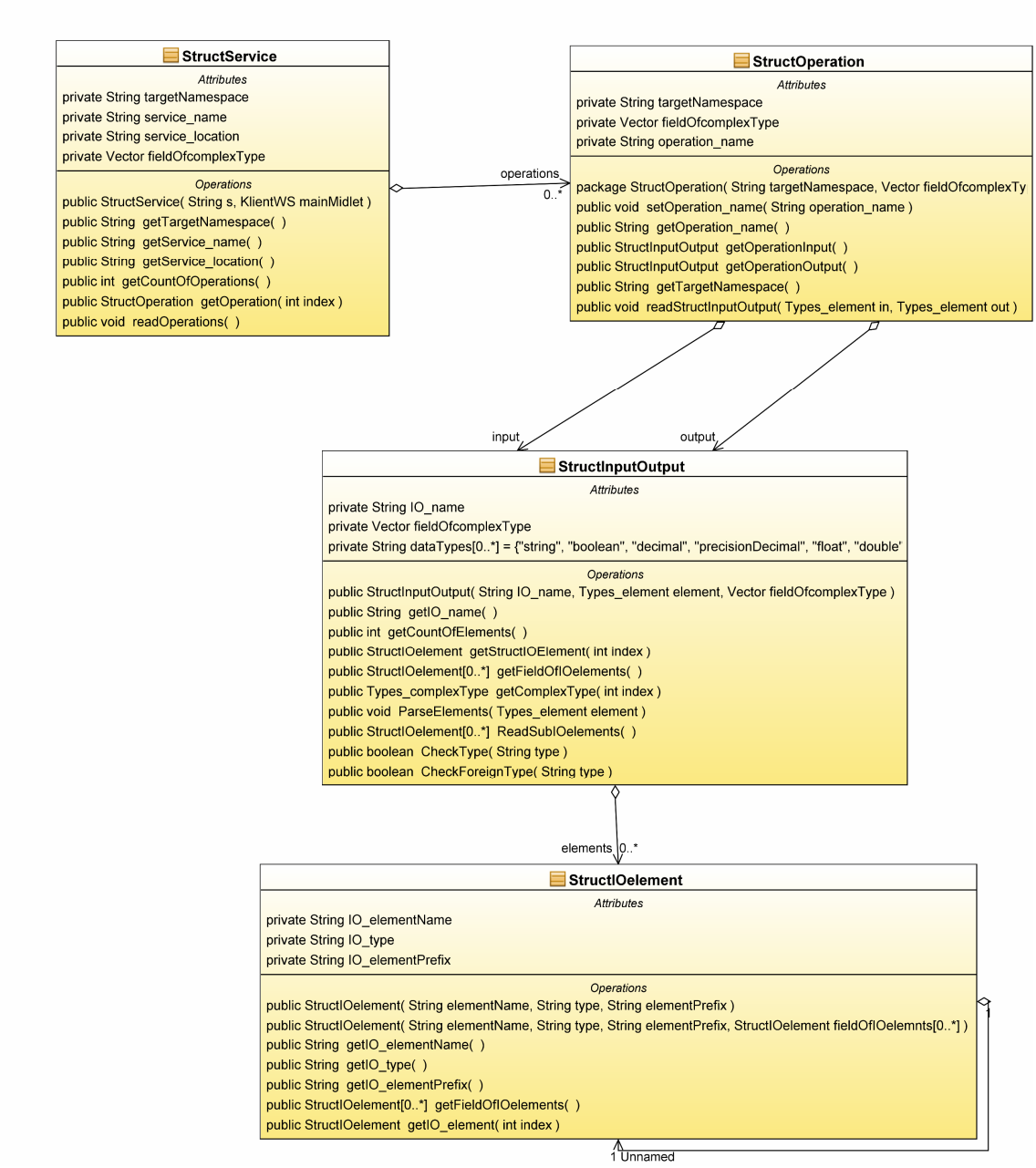
StructService.java – Koreňová trieda generovanej štruktúry prijíma vo svojom konštruktore jeden parameter, ktorým je textový reťazec obsahujúci WSDL dokument. V konštruktore potom nasleduje vytvorenie inštancie triedy `WSDL_parser`, ktorá zanalyzuje WSDL dokument a v pamäti vytvorí stromovú štruktúru získaných objektov. Ďalej sa v konštruktore nastavia súkromné premenné triedy ako meno služby, lokalizácia služby a menný priestor. Na záver je volaná metóda `readOperations()`, ktorá spustí generovanie štruktúry služby.

V metóde `readOperations()` sa postupne prechádza štruktúrou získanou procesom analýzy WSDL dokumentu od objektu typu `Service`, až po vnorené objekty typu `Types`, tak aby bola zachovaná korektnosť vstupných a výstupných dát jednotlivých operácii.
Výsledkom metódy je naplnenie poľa `operations` objektmi typu `StructOperation`. Trieda `StructOperation` je popísaná v nasledujúcom odstavci.

StructOperation.java – Hlavnou úlohou triedy `StructOperation` je, okrem inicializácie premennej `operation_name`, inicializácia premenných `input` a `output` typu `StructInputOutput` na základe dát prijatých v konštruktore triedy. Konštruktor prijíma okrem ostatných parametrov dva dôležité parametre: `Types_element input`, `Types_element output`. Tieto dva parametre reprezentujú podstromy štruktúry získané analýzou WSDL dokumentu, a nesú informácie o forme a obsahu požiadaviek a odpovedí SOAP komunikácie. Na základe týchto dvoch parametrov sa celá štruktúra služby rozdeľuje na dve časti. Jednu časť tvorí objekt `input` charakterizujúci formu vstupných dát operácie a druhú časť objekt `output` charakterizujúci formu výstupných dát.

StructInputOutput.java – Trieda `StructInputOutput` prijíma v konštruktore názov a príslušný vstupný, alebo výstupný podstrom WSDL dokumentu, ktoré sme spomenuli pri popise triedy `StructOperation.java`. Ďalšími parametrami sú vektory, ktoré obsahujú objekty pevne daných a komplexných dátových typov, ak sú prítomné v XML schéme elementu Types. Trieda obsahuje

metódu `parseElement(Types_element element)`, ktorej úlohou je naplnenie poľa `elements` objektmi typu `StructIOelement`. Objekty poľa získame z objektu element, ktorý metóda prijíma, ako parameter. Počas plnenia poľa metóda kontroluje dátový typ každého elementu XML schémy, a na základe toho inicializuje novú inštanciu triedy `StructIOelement`, pomocou troch konštruktorov. V prípade, ak je element základného dátového typu, vytvára inštanciu pomocou prvého konštruktoru triedy `StructIOelement`, ak je element pevne definovaného typu, vytvorí inštanciu podľa druhého konštruktoru a v prípade, že je element komplexného typu používa tretí konštruktor triedy, ktorý umožňuje zanorenie elementov (Obrázok 10).



Obrázok 9: Stromová štruktúra webovej služby, ktorá nesie informácie o službe nevyhnutné pre SOAP komunikáciu a procesy generovania GUI komunikácie.

```
public StructIOelement(String elementName, String type, String
                                                    elementPrefix)
{
     this.IO_elementName = elementName;
     this.IO_type = type.substring(elementPrefix.length()+1);
     this.IO_elementPrefix = elementPrefix;
     fieldOfIOelements = null;
     simpleTypeValues = null;
}

public StructIOelement(String elementName, String type, String
                              elementPrefix, String[] simpleTypeValues)
{
     this.IO_elementName = elementName;
     this.IO_type = type.substring(elementPrefix.length()+1);
     this.IO_elementPrefix = elementPrefix;
     fieldOfIOelements = null;
     this.simpleTypeValues = simpleTypeValues;
}

public StructIOelement(String elementName, String type, String
                     elementPrefix, StructIOelement[] fieldOfIOelemnts)
{
     this.IO_elementName = elementName;
     this.IO_type = type.substring(elementPrefix.length()+1);
     this.IO_elementPrefix = elementPrefix;
     this.fieldOfIOelements = fieldOfIOelemnts;
     simpleTypeValues = null;
}
```

Obrázok 10: Časť kódu znázorňujúca obidva konštruktory triedy `StructIOelement`.

StructIOelement.java – Trieda `StructIOelement` nesie informácie o každom z elementov, ktoré budú súčasťou SOAP komunikácie. Trieda obsahuje tri konštruktory, ktoré prijímajú rovnaké parametre s tým rozdielom, že druhý a tretí konštruktor prijímajú o jeden parameter viac. Pri elementoch základných dátových typov je volaný prvý konštruktor triedy a v situácii, keď XML schéma obsahuje elementy pevne definovaného typu, alebo komplexného typu, je volaný druhý, alebo tretí konštruktor. Prítomnosť elementov komplexného typu ošetruje práve rozširujúci parameter tretieho konštruktoru prijímajúci pole objektov typu `StructIOelement`. Prostredníctvom tohto poľa dosiahneme zanorenie objektov v štruktúre služby, reprezentujúcich elementy komplexného typu. Každý objekt triedy `StructIOelement` nesie informácie o názve elementu, dátovom type elementu, prefixe a v prípadne komplexných dátových typov obsahuje nenulové pole vnorených objektov. Na obrázku 9 je časť kódu triedy StructIOelement znázorňujúci obidva konštruktory.

5.2.5 Balíček gui_ws

Procesy generovania GUI určeného pre komunikáciu s webovou službou nie sú nijak zložité. Skupina obsahuje dva základné procesy. Úlohou prvého procesu je generovať GUI určené pre zadanie vstupných dát SOAP požiadavky a druhý proces generuje GUI pre reprezentáciu získaných dát zo SOAP odpovede. Obidva procesy sú implementované sa v triede `GUI_manager.java` balíčku `rus107.gui_ws` a sú reprezentované štyrmi hlavnými metódami triedy:

`FirstScreen()` – Metóda má za úlohu zobraziť, prostredníctvom komponentu List s možnosťou voľby, zoznam operácií , ktoré vybraná webová služba poskytuje. Ponuka, alebo zoznam služieb je riešený obdobne, ako hlavného menu aplikácie v balíčku `rus107.main_gui`. Po výbere operácie zo zoznamu je volaná metóda `Input_GUI(int numberOfOperation)`.

`Input_GUI(int numberOfOperation)` – Metóda ako parameter prijíma index vybranej operácie a na základe tohto indexu a štruktúry webovej služby, získanej predchádzajúcim procesom kapitoly 5.2.4, generuje vstupný formulár. Metóda využíva pomocné metódy triedy na zistenie a nastavenie obmedzení dátového typu textových polí formuláru. Tieto obmedzenia získa na základe špecifikácie XML schémy WSDL dokumentu webovej služby. Následne získané dáta z formuláru sú použité v procesoch SOAP komunikácie.

`ReadVstupneData()` – Metóda je volaná po výbere položky Odoslať v menu obrazovky so vstupným formulárom. Jej úlohou je získať vložené dáta z textových polí vstupného formuláru a pomocou tried SOAP komunikácie vytvoriť SOAP požiadavku. Následne je SOAP požiadavka odoslaná serveru webovej služby prostredníctvom metódy `Connect` HTTP komunikácie kapitoly 5.2.2. Metóda `Connect` vráti odpoveď zo serveru WS v podobe textového reťazca. Textový reťazec so SOAP odpoveďou je následne zadaný, ako parameter pri volaní metódy `Output_GUI(String strinSOAPresponse)`.

`Output_GUI(String stringSOAPresponse)` – Úlohou metódy je na základe prijatej SOAP odpovede serveru, ktorú prijíma ako parameter a pomocou triedy `SOAP_response.java` vytvoriť GUI reprezentujúce odpoveď na displeji mobilného telefónu. Výsledné GUI je tvorené komponentom `Form` naplneným komponentmi typu `TextField` v režime `UNEDITABLE`.

5.2.6 Balíček soap_parser

Skupinu procesov SOAP komunikácie je možné označiť aj ako SOAP parser. Procesy majú za úlohu generovať a analyzovať SOAP správy komunikácie s webovou službou. Procesy sú naimplemetované v dvoch triedach balíčku `rus107.soap_parser`:

SOAP_request.java – Trieda `SOAP_request.java` prijíma vo svojom konštruktore dva parametre. Jedným parametrom je časť štruktúry služby popísanej v podkapitole 5.2.4 a druhým parametrom je pole reťazcov (angl. `String`) obsahujúce dáta vložené vo vstupnom GUI formulára. Pomocou týchto dvoch parametrov trieda v metóde `Create_SOAPmessage()` vygeneruje SOAP požiadavku, ktorá bude odoslaná serveru poskytovateľa danej webovej služby.

SOAP_response.java – Úlohou triedy `SOAP_response.java` je syntaktická analýza dokumentu SOAP odpovede serveru WS. SOAP odpoveď prijíma trieda v podobe textového reťazca, ako jeden z parametrov konštruktoru. Druhým parametrom konštruktoru je časť štruktúry služby, ktorá nesie informácie o štruktúre spätnej SOAP odpovede serveru. Ďalej trieda obsahuje premennú typu `Vector` s názvom `responseTextFields`, do ktorej sa počas analýzy SOAP odpovede ukladajú elementy správy v podobe objektov typu TextField grafického užívateľského rozhrania. Vektor `responseTextFields` je volaný v metóde `Output_GUI(String stringSOAPresponse)`, ktorá naplní výsledný formulár GUI pomocou objektov typu `TextField`, ktoré sú obsahom vektoru.

6 Testovanie

Testovanie výslednej aplikácie generického klienta WS s dynamickým GUI môžeme rozdeliť do dvoch častí, reprezentovanými základnými procesmi verifikácie a validácie. Cieľom testovacieho procesu verifikácie je hľadanie nedostatkov v implementácii a úlohou procesu validácie je otestovať správnu funkcionalitu aplikácie. Aplikácia bola testovaná emulátorom Sun Java(TM) Wireless Toolkit 2.5.2_01 for CLDC vývojového prostredia NetBeans 6.7 a fyzickým zariadením bol mobilný telefón Nokia E52 s operačným systémom Symbian 60 Series.

Ako kritická vlastnosť výslednej aplikácie sa už pri návrhu javila rýchlosť chodu a zaťaženie procesoru mobilného zariadenia. Preto bol braný väčší dôraz na implementáciu procesov, ktoré značne ovplyvňovali práve rýchlosť aplikácie, konkrétne tým myslíme proces spracovania WSDL dokumentu a proces generovania štruktúry služby. Tieto dva procesy sú totiž časovo najnáročnejšie procesy, na ktorých stojí rýchlosť chodu celej aplikácie. Pri ich implementácii sme sa snažili striktne dodržiavať pravidlá uvedené v kapitole 3.1 pre dosiahnutie maximálnej efektivity aplikácie.

Druhým procesom testovania bola kontrola funkcionality aplikácie. Celý návrh i implementácia samozrejme smerovali k tomu, aby výsledná aplikácia bola schopná analyzovať zadané dokumenty WSDL, získané dáta interpretovať do internej štruktúry aplikácie a na základe týchto štruktúr generovať výsledné GUI a SOAP správy pre komunikáciu s WS. Pri testovaní však boli odhalené problematické miesta technológie WS, ktoré spôsobujú, že klienti WS disponujú určitými obmedzeniami voči kompletnej skupine WS. Počas testovania aplikácie sme dospeli k záveru, že problematickou časťou, určujúcou správnu funkcionalitu aplikácie sú WSDL dokumenty vytvárané poskytovateľmi WS. Napriek tomu, že popis WS je daný štandardom WSDL, existujú mnohé WS, ktorých popis nesplňuje úplnú špecifikáciu štandardu, prípadne obsahujú nefunkčné odkazy na externé XML schémy a pod. Prakticky je teda dosť problematické vytvoriť takého generického klienta, ktorý by dokázal odhaliť a spracovať všetky možné odchýlky nachádzajúce sa v popisoch WS.

Úlohou nami vytváraného klienta je práca s jednoduchými WS, čo bolo prakticky dosiahnuté. Aplikácia však kladie určité obmedzenia a nie je schopná spracovať všetky možné WS. Z tohto dôvodu môže zaiste nastať situácia, kedy užívateľ zvolí WS, s ktorú aplikácia nebude schopná spracovať. Klient pracuje s WS obsahujúcimi elementy základných dátových typov, pevne definovaných dátových typov (simpleType) a zložených dátové typov (complexType). V nasledujúcom texte sú vypísané príklady WS, ktoré boli použité pri testovaní a aplikácia ich podporuje a WS, ktoré nie sú aplikáciou podporované. U nepodporovaných WS sa nachádza v tabuľke dôvod nespolupráce.

Zoznam testovaných WS, ktoré boli podporované výslednou aplikáciou klienta WS:

Názov: Shakespeare
WSDL WS: http://www.xmlme.com/WSShakespeare.asmx?WSDL
Popis: Webová služba v anglickom jazyku umožňujúca vyhľadávanie citátov v Shakespearových dielach, ako napríklad: To be, or not to be.

Názov: Bible
WSDL WS: http://bible.petras-cz.eu/bible.asmx
Popis: Webová služba v českom jazyku, ktorá poskytuje verše a citáty z takmer všetkých svätých kníh.

Názov: HolidayWS
WSDL WS: http://www.holidaywebservice.com/Holidays/US/Dates/USHolidayDates .asmx?WSDL
Popis: Pomocou webovej služby je možné získať presné dátumy amerických štátnych sviatkov a iných významných dní.

Názov: Wheather
WSDL WS: http://ws.cdyne.com/WeatherWS/Weather.asmx?wsdl
Popis: Webová služba poskytuje aktuálne informácie o počasí v amerických štátoch podľa ZIP kódov.

Názov: VideoGamesFinder
WSDL WS: http://www.xmlme.com/WSVideoGames.asmx?WSDL
Popis: Webová služba na základe zadaného reťazca vyhľadá existujúcu viedohru dostupnú na trhu.

Názov: TemperatureConversions
WSDL WS: http://webservices.daehosting.com/services/TemperatureConversions.wso?WSDL
Popis: Webová služba umožňuje prevod stupňov medzi stupňami Celsia a Faradajovými.

Názov: CurrencyConvertor
WSDL WS: http://www.webservicex.net/CurrencyConvertor.asmx?WSDL
Popis: Webová služba poskytuje aktuálny kurz obrovského množstva mien celého sveta.

Názov: ISBNService
WSDL WS: http://webservices.daehosting.com/services/isbnservice.wso?WSDL
Popis: Webová služba umožňuje zistiť existenciu ISBN kódu knihy.

Názov: NumberConversion
WSDL WS: http://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL
Popis: Jednoduchá webová služba, ktorá prevádza čísla na textové reťazce. WS je v anglickom jazyku.

Názov: Euro2008
WSDL WS: http://euro2008.dataaccess.eu/footballpoolwebservice.wso?WSDL
Popis: Webová služba poskytuje rôzne informácie o futbalovom turnaji Euro 2008, ako napríklad: názvy futbalové štadiónov, názvy zúčastnených tímov a pod.

Názov: GeoIPService
WSDL WS: http://www.webservicex.net/geoipservice.asmx?WSDL
Popis: Webová služba poskytuje operácie pre prácu s IP adresami. Pomocou WS je možné napríklad zistiť svoju IP adresu, prípadne miesto výskytu iných IP adries.

Názov: SunSetRiseService
WSDL WS: http://www.webservicex.net/MortgageIndex.asmx?WSDL
Popis: Webová služba poskytuje čas východu a západu slnka, kdekoľvek na zemi.

Názov: MortgageIndex
WSDL WS: http://www.webservicex.net/MortgageIndex.asmx?WSDL
Popis: Webová služba poskytuje prácu s indexom mortality, pravdepodobne oblasti spojených štátov amerických.

Názov: FedWire
WSDL WS: http://www.webservicex.net/FedWire.asmx?WSDL
Popis: Webová služba vyhľadáva Fedwire účastníkov na základe kontaktných údajov ako meno, lokalizácia apod.

Zoznam testovaných WS, ktoré neboli podporované výslednou aplikáciou klienta WS:

Názov: Lotto
WSDL WS: http://reto.checkit.ch/Scripts/Lotto.dll/wsdl/IgetNumbers
Dôvod: Webová služba fyzicky neobsahuje element Types s XML schémou dokumentu, ani odkaz s URL adresou externého dokumentu XML schémy. Nie je preto možné zistiť formát SOAP správ komunikácie s webovou službou.

Názov: ChangeRates
WSDL WS: http://www.csas.cz/csws/changerates/ChangeRates.jws?WSDL
Dôvod: Popis webovej služby obsahuje neštandardný prefix základných elementov štandardu WSDL.

Názov: FreeIBANValidate
WSDL WS: http://www.unifiedsoftware.co.uk/freeibanvalidate.wsdl
Dôvod: Webová služba rovnako neobsahuje element Tzpes s XML schémou ani odkaz s URL adresou externého dokumentu XML schémy.

---

7 Záver

Hlavným cieľom bakalárskej práce bolo oboznámenie sa s technológiou webových služieb, vývojovou platformou Java ME a možnosťami práce s XML dokumentmi v aplikáciách. Výsledkom práce je funkčná aplikácia Klienta WS určená pre mobilné zariadenia pracujúca s jednoduchými webovými službami. Spustenie aplikácie vyžaduje mobilné zariadenie s podporou Java ME s verziou profilu MIDP - 2.0.

Nakoľko sú webové služby technológiou s naozaj veľmi širokou problematikou, poskytujú aj veľké množstvo vlastných variácií. Touto vetou mám na mysli hlavne popis webových služieb prostredníctvom dokumentov štandardu WSDL. Každý WSDL dokument je v podstate jedinečným dokumentom s vlastnou úrovňou zložitosti, ktorá v konečnom dôsledku určuje aj zložitosť webovej služby. To znamená, že každá webová služba poskytuje a pracuje s dátami rôznej štruktúry, zložitosti a obsahu. Vytvorená aplikácia klienta WS pracuje s jednoduchými webovými službami, z tohto dôvodu kladie určité obmedzenia voči komplikovanejším webovým službám, ako môžu byť napríklad služby s vlastným definovaným menným priestorom a pod. Nevidím však problém v ďalšom rozšírení aplikácie o podporu zložitejších webových služieb, ktoré by predstavovalo podrobnejšie prepracovanie analýzy dokumentov WSDL. WSDL popisy totiž obsahujú naozaj veľké množstvo informácii, ktoré je možné z nich „vydolovať“ a použiť v aplikácii. Ako jedným z rozšírení aplikácie vidím práve komplexné využitie informácii z WSDL dokumentov. Po tomto rozšírení analýzy WSDL a doimplementovaní jednoduchého rozhrania by bolo možné kód aplikácie, konkrétne balíčky wsdl_parser a service, využívať ako univerzálnu knižnicu poskytujúcu základné informácie o WS. Takáto knižnicu by našla určite uplatnenie v ďalších projektoch využívajúcich WS.

Vďaka tejto bakalárskej práci som si rozšíril svoje znalosti a získal cenné skúsenosti z oblasti technológií webových služieb a práce s XML dokumentmi v mobilných aplikáciách. Na záver by som chcel len dodať, že webové služby sú naozaj veľmi prínosnou a praktickou technológiou, hlavne v kombinácii použitia generického klienta WS a mobilného zariadenia. Táto kombinácia totiž poskytuje užívateľovi praktické a rýchle získanie potrebných informácii na akomkoľvek mieste s dostupným signálom mobilného operátora. Ako jedinú nevýhodu webových služieb dnes, vidím nízky počet poskytovaných služieb v českom a slovenskom jazyku zameraných na problematiku nášho sveta.

Zoznam použitej literatúry

[1] Michael J.Young, *XML krok za krokem*, 2. vydaní Brno: Computer Press, a.s., 2006, 472 s., ISBN: 80-251-1070-2, s. 179

[2] Cajthaml Martin, *Co Vám přináší webové služby?* [online], 2006 [cit. 2010-02-19], Dostupné z: <http://www.symbio.cz/clanky/co-vam-prinasi-webove-sluzby.html>

[3] *SOAP Tutorial* [online], 2010 [cit. 2010-02-22], Dostupné z: <http://www.w3schools.com/soap/default.asp>

[4] Kosek J., *Využití webových služeb a protokolu SOAP při komunikaci* [online], 2002 [cit. 2010-02-28], Dostupné z: < http://www.kosek.cz/diplomka/html/websluzby.html>

[5] *WSDL Tutorial* [online], 2010 [cit. 2010-02-28], Dostupné z: <http://www.w3schools.com/soap/default.asp>

[6] Kotásek M., *Bakalářská práce: Protokoly služeb Internetu* [online], [cit. 2010-02-28], Dostupné z: <http://www.cs.vsb.cz/grygarek/POS/kotasek/htp02.htm>

[7] *Universal Description Discovery and Integration* [online]. Wikipedia, 2010 [cit. 2010-02-28], Dostupné z: <http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration>

[8] Moravec P., *Programming strategies* [online], 2010 [cit. 2010-03-16], Dostupné z: < http://wiki.cs.vsb.cz/images/4/4d/TAMZ-L10.pdf>

[9] Topley K., *J2ME v kostce*, Brno: Grada Publishing, a.s., 2004, 499 s., ISBN: 80-247-0426-9, s. 86

[10] Procházka J., *J2ME pro pokročilé - XML* [online], 2003 [cit. 2010-04-01], Dostupné z: <http://interval.cz/clanky/j2me-pro-pokrocile-xml/>

Zoznam príloh

Príloha A – Užívateľská príloha
Príloha B – Diagramy tried
Príloha C – Priložené CD (inštalačné súbory, zdrojové kódy aplikácie, programátorská príručka)

A Príloha – Užívateľská príručka

A.1 Inštalácia aplikácie

Inštalácia Java aplikácií na mobilné zariadenia je všeobecne veľmi jednoduchá. V podstate ide o import a spustenie jedného zo súborov s príponou .jar, alebo .jad. Po spustení sa zaháji automatická inštalácia. Na priloženom CD nájdete súbory s názvami KlientWS.jar a KlientWS.jad potrebné k inštalácii klienta WS s dynamickým GUI pre mobilné telefóny. Jeden zo súborov importujte do mobilného zariadenia, prostredníctvom rozhrania Bluetooth, prípadne pomocou USB kábla.
V mobilnom zariadení stačí importovaný súbor otvoriť a spustí sa inštalácia aplikácie. Pri prvom spustení sa Vás aplikácia opýta, či chcete importovať ukážkové príklady webových služieb. Pre prvé otestovanie aplikácie odporúčam import ukážkových príkladov WS povoliť.

A.2 Základné menu aplikácie

Po spustení aplikácie sa objaví základné menu, ktoré obsahuje 5 položiek:

Spusti WS – Prvá položka menu umožňujúca spustenie WS. Podrobnejší popis práce s položkou bude popísaný v ďalšom bode príručky.

Databáza WS – Ponuka druhej položky menu umožňuje správu webových služieb uložených v trvalom pamäťovom priestore mobilného zariadenia. Rovnako i možnosti správy WS budú popísané v ďalších bodoch príručky.

O aplikácii – Položka menu obsahuje základné informácie o aplikácii, ako názov aplikácie, autora, verziu, účel a krátku charakteristiku aplikácie.

Pomocník – Pod štvrtou položkou menu sa nachádza skrátená verzia užívateľskej príručky. Pomocník je riešený formou zoznamu, ktorého položky reprezentujú názvy jednotlivých situácií, ktoré by mohol mať užívateľ v pláne vyriešiť. Pod jednotlivými položkami sa nachádza krátke riešenie problému.

Koniec – Posledná položka menu ukončuje a uzatvára aplikáciu.

A.3 Výber a spustenie webovej služby

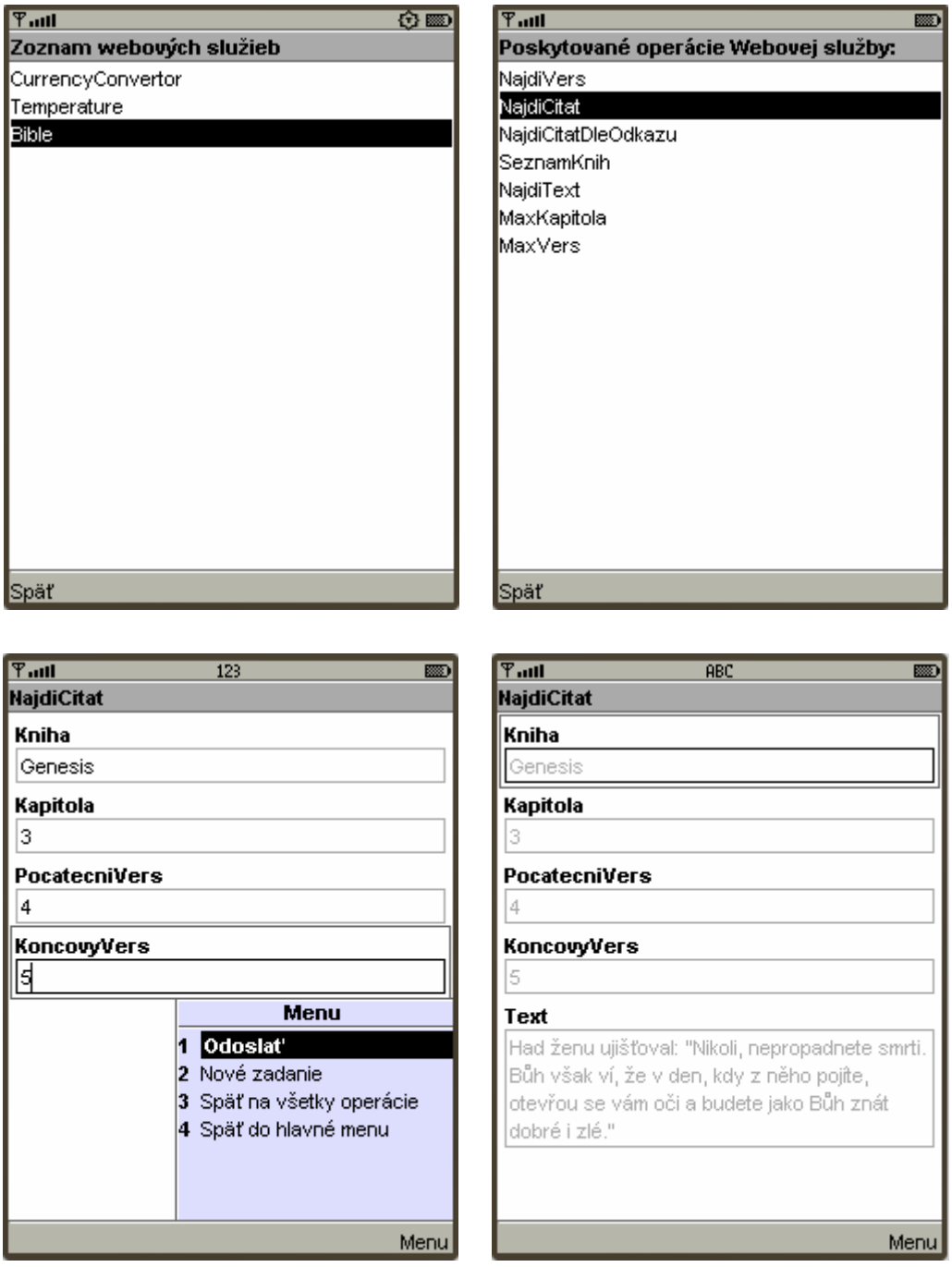
Výber a spustenie webovej služby uskutočníte pod položkou Spusti WS hlavného menu. Po výbere tejto položky hlavného menu sa Vám na obrazovke zobrazia uložené webové služby v trvalom pamäťovom priestore mobilného telefónu. Ak ste spustili aplikáciu úplne prvý krát a nepovolili ste import ukážkových webových služieb, bude zoznam prázdny. Vtedy si musíte vložiť webové služby sami. Postup, ako na to nájdete v ďalšom bode príručky – A.3 Správa webových služieb. Druhý prípad nastane vtedy, ak ste pri prvom spustení povolili import ukážkových príkladov WS. V tomto prípade sa Vám na obrazovke zobrazí zoznam obsahujúci tri položky reprezentujúce jednotlivé WS. Do aplikácie môžete vkladať nové WS, prípadne vymazávať a upravovať už uložené WS. Spôsob, ako tieto činnosti prevádzať sa dozviete v ďalšom bode príručky.
Zobrazený zoznam WS na obrazovke je interaktívny. To znamená, že po výbere WS a potvrdení stredným (potvrdzovacím) tlačidlom mobilného telefónu sa spustí vybraná webová služba.

Po spustení webovej služby sa spustia vnútorné procesy aplikácie, ktoré môžu nejakú dobu trvať v závislosti na mobilnom zariadení a internetovom pripojení. Po vnútornom spracovaní WS aplikácia zobrazí zoznam operácií, ktoré WS poskytuje. Vyberiete si jednu z operácií a potvrdíte stredným (potvrdzovacím) tlačidlom. Po výbere operácie Vám aplikácia zobrazí formulár určený pre zadanie vstupných dát, ktoré budú následne odoslané serveru webovej služby. Starostlivo vyplňte všetky polia formulára a z menu pod jedným z funkčných tlačidiel vyberte možnosť Odoslať. Požiadavka bude odoslaná serveru WS, ktorý ho spracuje a vráti spätnú odpoveď Vášmu klientovi. Klient výslednú správu zobrazí na obrazovke mobilného zariadenia.

Obrazovky formulárov, ktoré reprezentujú komunikáciu s WS obsahujú v jednom s funkčných tlačidiel menu, ktoré umožňuje pohyb v rámci vybranej webovej služby, prípadne návrat do hlavného menu. Menu môže obsahovať nasledujúce položky:

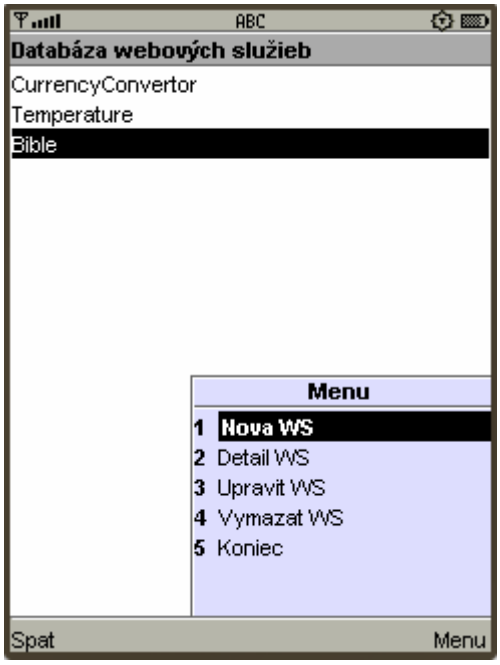
- Odoslať
- Nové zadanie
- Späť na všetky operácie
- Späť do hlavného menu
- Koniec

Ukážkový priebeh výberu a spustenia WS môžete vidieť na nasledujúcich obrazovkách:



A.4 Správa webových služieb v pamäti mobilného zariadenia

Pod druhou položkou Databáza WS základného menu sa zobrazí na obrazovke opäť zoznam WS uložených v trvalom pamäťovom priestore mobilného zariadenia. V tejto časti aplikácie však nespúšťame WS, ale máme k dispozícii akúsi správu databázy WS. Pod jedným z funkčných tlačidiel sa nachádza príslušné menu, ktoré nám ponúka možnosti práce s vybranou WS. Menu obsahuje nasledujúce položky:

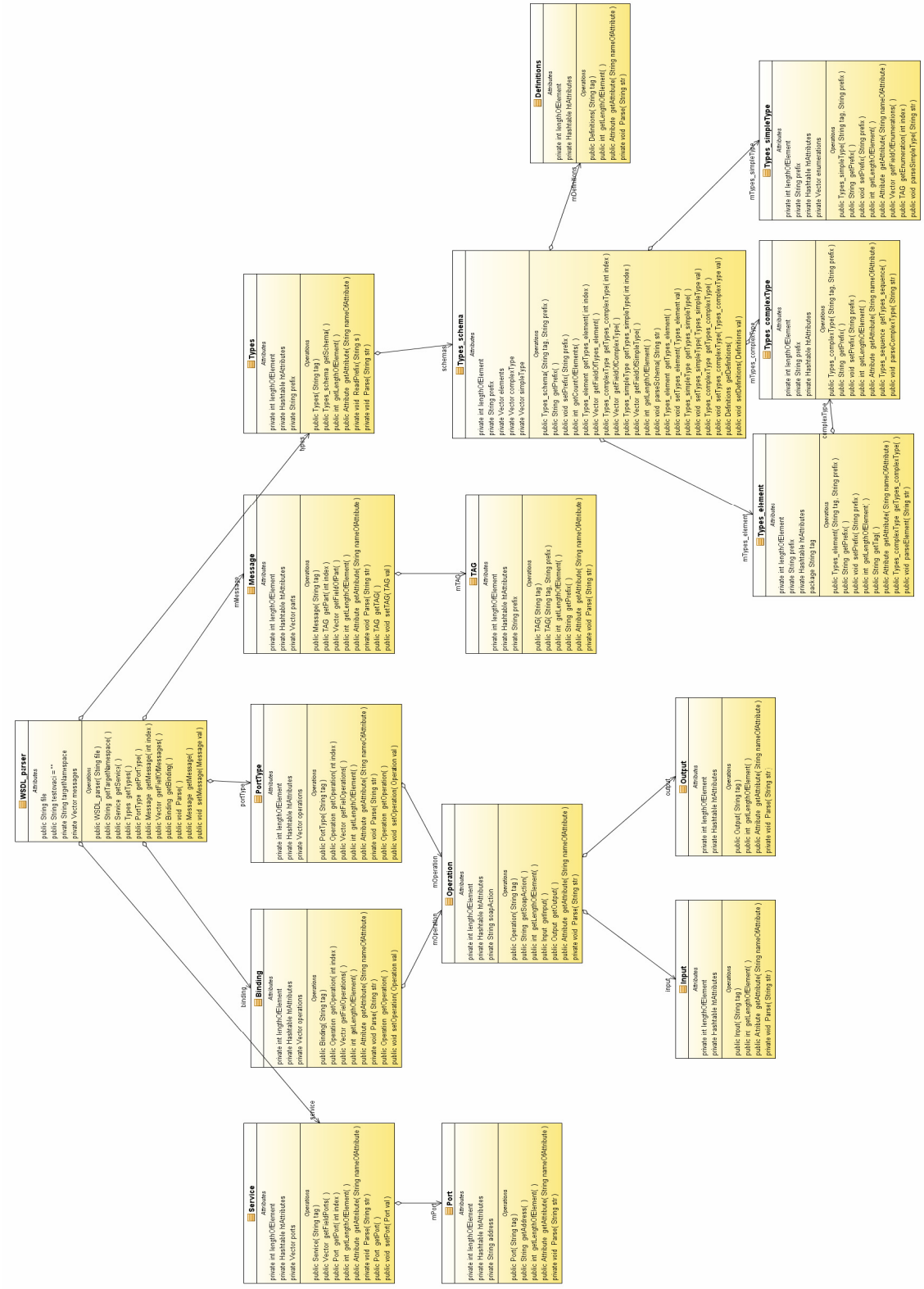


Nová WS – Položka menu Nová WS umožňuje pridať novú WS do databázy. Po výbere položky sa zobrazí formulár s poľami pre zadanie názvu a URL adresy WSDL dokumentu WS.
Detail WS – Položka menu Detail WS zobrazí na obrazovke detailný výpis webovej služby.
Upraviť WS – Po výbere položky Upraviť WS sa zobrazí záznam WS na obrazovke vo formáte editácie. Názov a URL adresu je možné upraviť a uložiť späť do mobilného zariadenia.
Vymazať WS – Položka umožňuje trvalé odstránenie WS s pamäti mobilného zariadenia.
Späť – Položka vracia užívateľa do hlavného menu aplikácie.

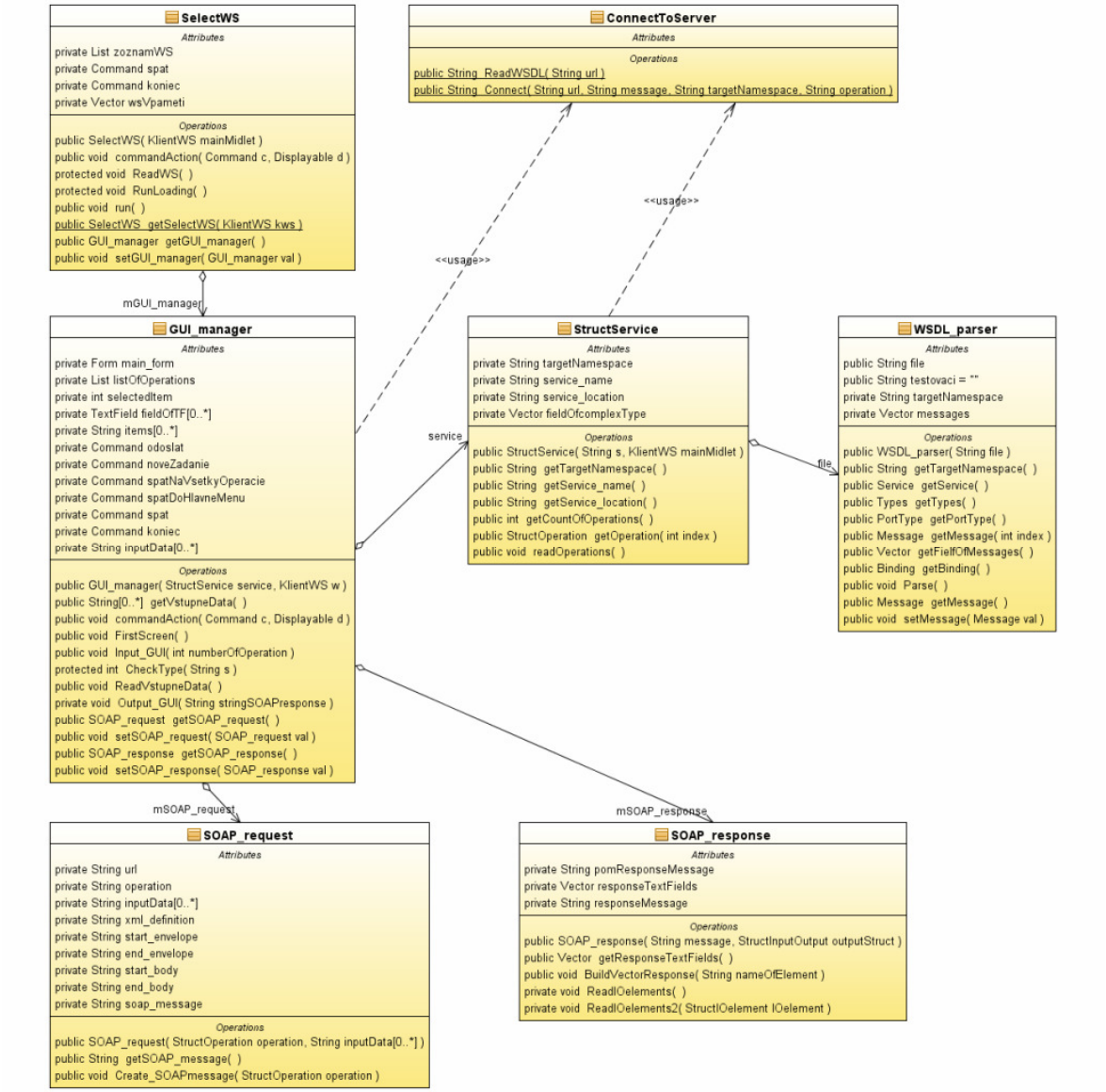
Obrazovky zobrazené po výbere položiek Nová WS, Detail WS a Upraviť WS sú takmer identické. Hlavný rozdiel je v ich následnej vnútornej reprezentácii. Ukážky sú na nasledujúcich obrázkoch:

B Príloha – Diagramy tried

B.1 Diagram tried balíčku rus107.wsdl_parser



B.2 Diagram tried, ktoré sa podieľajú na spracovaní a prezentácii WS



B.3 Diagram tried hlavného GUI aplikácie